COMPUTER SYSTEMS LABORATORY

STANFORD UNIVERSITY · STANFORD, CA 94305-4055

LELAND STANFORD JUNIOR UNIVERSITY · ORGANIZED 1891 ·

1

Multi-Dimensional Data Compression for Portable Communication

Technical R&D Status Report
July 1995 - December 1995

Principal Investigator
Teresa H. Meng

Department of Electrical Engineering
Computer Systems Laboratory
Stanford University
meng@tilden.stanford.edu

Sponsored by Advanced Research Projects Agency
Monitored by the United States Department of the Army
Contract Number: DABT-91-K-0002
Effective Date of Contract: August 1, 1991
Contract Expiration Date: December 31, 1994

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
3/27/95

3. REPORT TYPE AND DATES COVERED
Final Semi-Annual, 8/1/91-12/31/94

4. TITLE AND SUBTITLE
Multi-Dimensional Data Compression for Portable Communication

5. FUNDING NUMBERS
DABT63-91-K-0002

6. AUTHOR(S)
Teresa H. Meng

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Stanford University
Electrical Engineering Dept., CSL
CIS 132, M/C4070
Stanford, CA 94305-4070

8. PERFORMING ORGANIZATION REPORT NUMBER
--

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
ARPA
ATTN: ISTO
3701 North Fairfax Drive
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
See attached report

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UL	UL	UL	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities.
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.
DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA - Leave blank.
NTIS - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Multi-dimensional Data Compression for Portable Communication

1.0 Overview and Summary

This project provides an integrated design approach of a portable video-on-demand system capable of delivering high-quality image and video in a wireless communication environment. The research focuses on both the algorithm and circuit design techniques developed for implementing a low-power video compression/decompression system at power levels that are two orders of magnitude below existing solutions. The three main technological challenges in realizing such a system are high compression efficiency for transmission bandwidth reduction, error resiliency to minimize the effects of channel error, and low-power implementation for portability.

The required power reduction can best be attained through recasting compression algorithms for energy conservation. We developed compression algorithms with a compression efficiency similar to or better than industry standards but requiring minimal computation energy in their hardware implementations. Examples of algorithmic trade-offs are the development of a vector quantization scheme that allows on-chip computation to eliminate off-chip memory accesses, the use of channel-optimized data representations to avoid the error control hardware that would otherwise be necessary, and the coding of internal data representations to further reduce the energy consumed in on-chip data exchanges. The architectural and circuit design techniques used include the selection of a filter bank structure that minimizes the energy consumed in datapath, the data shuffle strategy that results in reduced internal memory size, and the design of digital and analog circuits optimized for low supply voltages.

In our portable video decoder design, more than a factor of 100 reduction in power consumption is demonstrated. We achieved this high level of power efficiency without degrading performance in video quality. In addition, by embedding error resiliency into our system development consideration, we were able to avoid the hardware complexity and bandwidth penalty incurred by the use of error-correcting codes.

Our hardware prototype is a portable video-on-demand system for compressing/decompressing full-motion video transmitted through a wireless link at 500 *Kbits/s*ec to 1 *Mbits/s*ec range. This system includes a portable low-power video decoder module with a color display and a real-time encoder base station implemented on a DSP multiprocessor. In this report we will describe the design trade-offs of the prototype for low-power purposes, and quantify the system's performance in both compression efficiency and power dissipation.

On the software design, we have developed an integrated software package for compression algorithm development, including a library of compression modules, tools for calculating the relevant performance figures, and a database of image/video sequences. These modules and tool boxes can be used as basic cells in a compression algorithm assembler with which new algorithms can be easily constructed and compared with existing ones on the same video data set. This compression software package has been distributed to more than 15,000 users with approximately 60,000 transfer accesses in the past year.

1.1Major Accomplishments

This section summarizes the major accomplishments of this project:

Design and fabrication of the video decoder chip set - We completed and fabricated the low-power video decoder chip set, which includes a subband decoder chip, a pyramid-vector-quantization (PVQ) decoder chip, and a low-power D/A converter (DAC) for the control of a color LCD display. The low-power video decoder chip set capable of decoding 30 *frames/sec* of video at a power level of 8 *mW*, 100 times below existing video decoding chips, was designed and tested. Extensive simulations and measurements have been performed to verify its correct operation. First a C program generated bit-level input and output data files based upon real image sequences. These data files were run through a Verilog model of the decoder chips for bit-level verification. The Verilog model was then used to generate input vectors and output checks for the switch-level simulation of the chip layout. A test board was also fabricated to allow for real-time testing of the decoding operation. Both chips have been tested and successfully worked together in tandem.

Completion of the portable video-on-demand prototype system - The portable video-on-demand prototype system has also been completed, which includes a portable video decoder module and a real-time encoder base station implemented on a DSP multiprocessor. The portable decoder and the stationary encoder communicate through a wireless link of spread spectrum radios. Altera FPGAs were programmed to control the radios and data transfers between the encoder station and the portable decoder module. Radio transmission at 727 *Kbits/sec* has been tested to be fully functional. The video decoder board, which contains our video decoder chip set and a color LCD display, has been tested to be fully functional.

Development of error-resilient compression algorithm - An error-resilient compression algorithm exceeding the performance of standard image compression algorithms such as JPEG has been developed and finalized. This algorithm outperforms JPEG at all bit rates of interest, without requiring the use of variable-rate entropy codes. In addition to this high degree of compression efficiency, our compression algorithm embeds error-resiliency in the coding process, guaranteeing consistent video quality under all error conditions without the use of error-correcting codes. Its hardware complexity is less than half of that of the JPEG-like algorithms, allowing for a low-power implementation without the need of any external hardware support such as frame buffers and video control circuitry.

Completion of an software package for compression algorithm development - We have developed an integrated software package for compression algorithm development, including a library of compression modules, tools for calculating the relevant performance figures, and a database of image/video sequences. These modules and tool boxes can be used as basic cells in a compression algorithm assembler with which new algorithms can be easily constructed and compared with existing ones on the same video data set. This compression software package has been distributed to more than 15,000 users with 52,468 transfer accesses in the past year.

Technology Transfer and Commercialization - The Stanford Technology Licensing Office initiated and completed the patent and licensing process for the commercialization of the results of this project. The technology has been introduced to more than 20 companies around the world by the Technology Licensing Office and discussions with a number of these companies are underway at this time.

Technology Transfer to IBM - IBM has transferred our low-power video decoder as a test vehicle for their SOI technology. As our decoder chips have been fabricated in CMOS technology, a similar design in

the SOI technology will not only provide an opportunity to test the technology limits, but also deliver a quantitative measure of the amount of performance advantage attainable by using SOI over standard CMOS. As a start, we have translated the subband decoder chip for IBM's design rules and taped out the design to IBM for SOI fabrication.

Development of psychovisual-based scalable compression techniques - We have developed a psycho-visual based scalable compression algorithm that allows adaptive rate control to accommodate different bandwidths available in an open network. This algorithm will be applied to distributing Stanford televised lectures to the Internet, where video data will be transmitted and decoded in real time based on the instantaneous bandwidth available to each user on the net. The compression performance evaluation is based on our earlier work on psycho-visual distortion measure. We have also designed and implemented an end-to-end software only scalable video delivery system to demonstrate the effectiveness of this algorithm.

Minimization of metastability in interface circuits - For the design of low-power interface circuits such as A/D converters and synchronous-asynchronous communication, we have improved and characterized the effects of loading, supply voltages, and technology scaling on the metastable parameters of CMOS latches. To demonstrate our scheme, we have designed and tested a low-power 6-bit 80 *MHz* CMOS flash A/D converter, whose metastable error rate was measured to be six orders of magnitude below that of other A/D converters.

2.0 Portable Video Decoder Prototype

We will first discuss the prototype of our portable video decoder module, as shown in Figure 1. The compression algorithm used in this module is reviewed in Appendix A, which comprises subband decomposition followed by pyramid vector quantization (PVQ). The subband/PVQ encoded bit stream is transmitted through a wireless link with a bandwidth of 727 *Kbps* to the video decoder module. The received bit stream is decoded by the PVQ decoder chip into subband coefficients. The subband decoder chip then filters these coefficients and reconstructs them into images represented as three digital RGB color components. The digital RGB color components are converted to an analog format by a D/A converter (DAC) suitable for the color LCD display. The PVQ decoder chip, the subband decoder chip, and the low-power DAC were all custom-designed and form our decoder chip set. The controller for the LCD display and video synchronization signals are generated by the subband decoder chip. For global synchronization, the entire portable video decoder runs on a common clock, with the subband decoder chip at half the rate of the PVQ decoder chip.

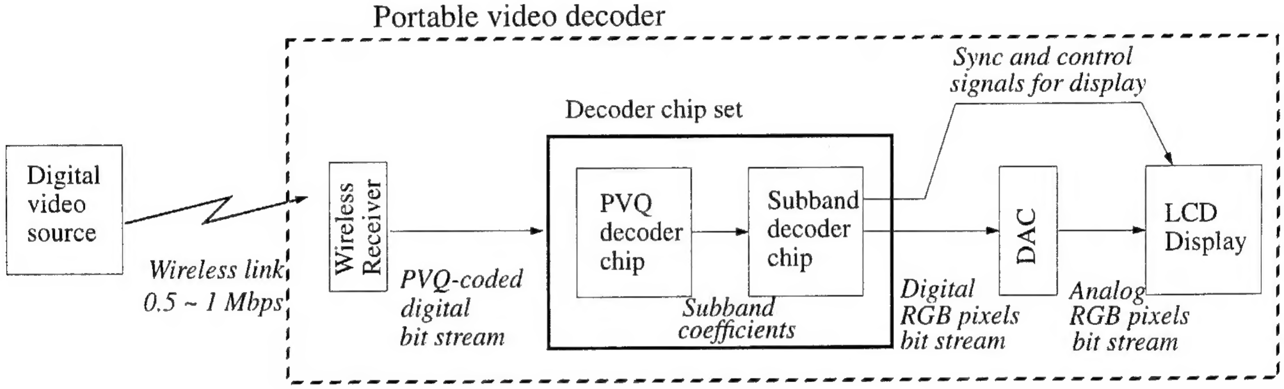


Figure 1. System organization of the portable video decoder.

In comparison with the C-Cube JPEG decoder, implemented in 1.2 μm COMS technology dissipating 2 W in decoding 30 frames of video per second [8], our subband/PVQ decoder is more than 100 times more power efficient, not accounting the power dissipated in accessing off-chip memory necessary in the JPEG decoding operation. Within this factor of 100, a factor of 10 can be easily obtained by voltage scaling of the power supply. Reduced supply voltage, however, increases circuit delay. This increase in delay needs to be compensated for by duplicating hardware, or chip area, to maintain the same real-time throughput [1]. The fact that our decoder module consists of only two custom chips, without requiring any off-chip memory support, indicates the simplicity of our decoding operation, one of the established goals of our compression algorithm design. How we achieved the other factor of 10 reduction in power is one of the focuses of this project.

2.1 Subband Decoder Chip

In designing the subband decoder, we emphasized a low-power implementation without introducing noticeable degradation in decompressed video quality. As memory accessing is by far the most power-consuming operation, the main design strategy has been to eliminate memory accesses in favor of on-chip

computation. A two-dimensional subband decoder has been designed for real-time video decompression in low-power applications. The chip dissipates less than 1.2 mW at a 1 V supply, delivering subband decomposition at 1.3 Mpixels/sec, for display of 176 pixels wide, 240 lines, and 30 frames per second color video. The chip is capable of reconstructing 4 levels of hierarchical subband structures for images up to 256 pixels wide and requires no external hardware support such as frame buffers or video control.

For portable applications with multi-media capabilities, real-time video decompression with extremely low power levels is needed to extend battery life. The single-chip subband decoder described in this section implements the real-time subband decompression used in many compression algorithms [2] with minimum power consumption. Furthermore, this single-chip decoder does not require any external hardware support such as off-chip memory or video control for the delivery of 30 frames per second of video signals to a color display.

2.1.1 Subband Coding

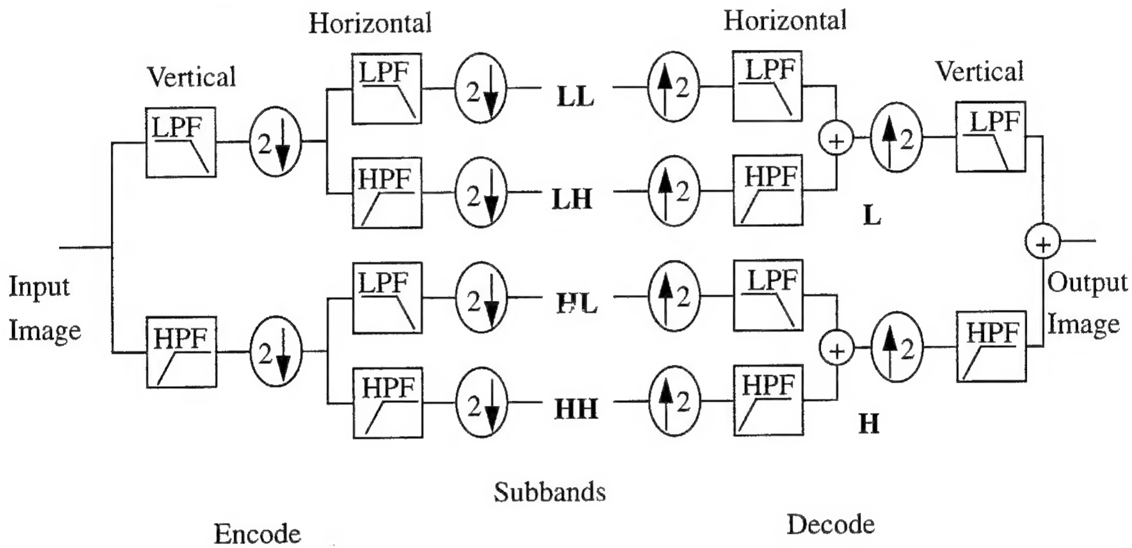


Figure 2. Two-dimensional subband filtering.

Subband encoding, shown in Figure 2, filters and decimates the image data with a high- and low-pass filter in the horizontal and vertical direction. This produces four output bands: low vertical/low horizontal (LL), low vertical/high horizontal (LH), high vertical/low horizontal (HL), and high vertical/high horizontal (HH) [2]. The LL band, containing most of the image information, is re-filtered creating another level of 4 subbands. Figure 2 illustrates the final band configuration after the luminance component (Y) is filtered through four levels, forming 13 bands. The two chrominance components (U and V) are filtered through three levels, forming 10 bands each. The decoding process recursively reconstructs LL bands from their respective lower-level bands. Our chip implements this decoding process and supports multiple levels of subband decomposition necessary for higher compression performance.

For a more detailed description of our compression algorithm, please refer to Appendix A.

2.1.2 Chip Architecture

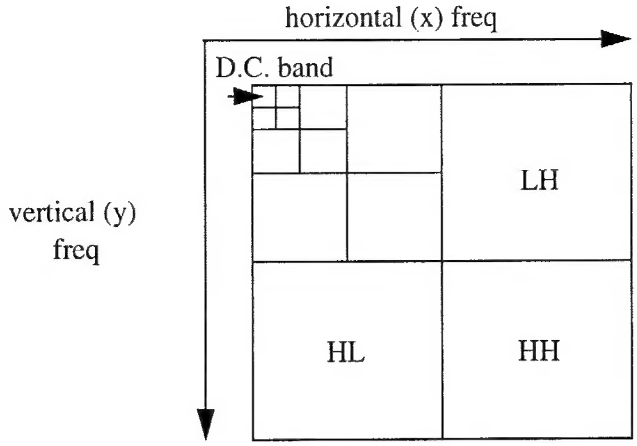


Figure 3. 4-level subband decomposition.

For low power, the design exploits the natural parallelism of the subband algorithm to achieve high peak performance, providing excess throughput that can be traded off for lower power by reducing the supply voltage. To further reduce the total decoding power, low-power design techniques and memory-efficient architectures were utilized to eliminate the need for off-chip memory support, removing the high power consumption of external memory access and board level I/O. The decoder also has built-in color conversion and video display control, providing a single-chip solution for low-power video-rate subband decompression.

The overall chip architecture including the filter operation is shown in Figure 3. To reduce datapath power consumption and the internal memory size with no degradation in compression performance, extensive simulation was used to determine the filter coefficients. A four-tap asymmetric wavelet filter, (3,6,2,-1), was chosen for the low-pass filter. Only one 3-2 adder with hardwired shifts performs the upsampling and filtering operation. The same hardware also implements the high-pass filter, (1,2,6,-3), by reversing the coefficients and negating the (6,-1) pair.

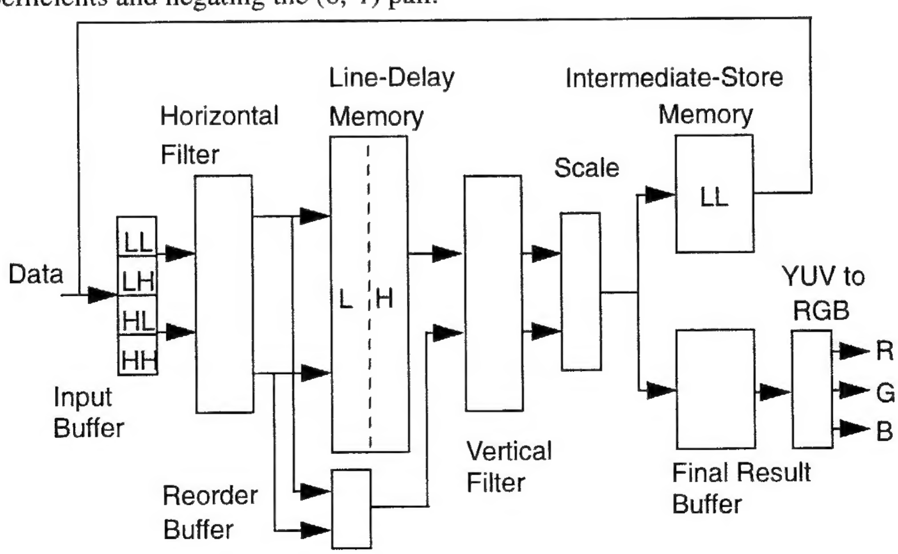
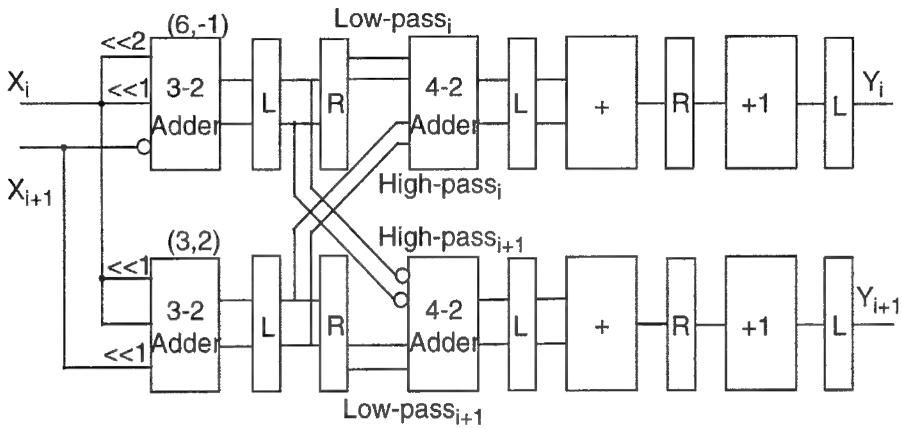


Figure 4. Subband decoder architecture.

The two pairs of filter coefficients, (3,2) and (6,-1), are simultaneously applied to two horizontally consecutive inputs from the same band, fetched from the input buffer. This produces, as seen in Figure 4, two horizontal results every two cycles creating one line of vertical low-passed (L) and high-passed (H) data (defined in Figure 1). These results are rounded down to the input precision of 10 bits to reduce the amount of memory storage required, which achieves the same signal to quantization noise ratio as truncating to 12 bits.



X: LL,LH,HL or HH
Y: L or H
i: horizontal index
L: Latch
R: Register (master/slave latch)
<<: left shift

Figure 5. Subband filtering implementation.

The vertical filter uses the filtered results from the previous line held in a line-delay memory. The vertical filter, operating similarly as the horizontal filter, generates two vertical reconstructed outputs every two cycles. After filtering, another shift/add operation performs unity gain normalization. Then an intermediate-store memory transposes the vertical results to allow power efficient horizontal access.

The intermediate-store memory, totalling 10 Kbits (128x4x20), holds two lines of subband coefficients for each level and then feeds them back to the input buffer for processing of the next level. The order of subband levels is interleaved such that both lines of outputs for a given level are consumed before new results are generated, thereby requiring only two lines of storage for the intermediate-store memory and one line for the line-delay memory. This interleaving scheme reduces the total amount of memory required by over a factor of 4, thus both saving on-chip memory accessing power and eliminating the need for off-chip memory support.

When the top level of subband decomposition is reconstructed, the resulting image data is put into the final result buffer which holds four lines of data up to a maximum of 256 pixels each. The video controller module determines when to read from the output buffer, sending the YUV values to the color conversion datapath. Shift/adds implement the YUV to RGB conversion that requires only one 3-2 and 5 full adds per output pixel. Gated clocks were implemented whenever possible so that during video blanking periods, only the video controller runs while the datapath and memory hold their current state.

When processing the highest frequency bands, the design takes advantage of the large percentage of zeros in these bands. The data is zero run-length encoded, reducing the number of external inputs by almost a factor of 4, lowering the overall system power by reducing the power dissipated in the external input source. Additionally, the control logic detects when zero values are present and skips their horizontal

processing. As a result, the average number of horizontal operations per output pixel, including all intermediate level and final image data for all three components, is reduced from 1.98 to 1.23, resulting in a 15% measured reduction of total chip power.

2.1.3 Power Performance

Figure 6(a) illustrates the power dissipation at the maximum operating frequency for various supply voltages. Figure 6(b) illustrates the variation in energy and delay as a function of the supply voltage. At a supply voltage of 1 *V*, the decoder chip operates at the required 3.2 *MHz* real-time video rate and dissipates under 1.2 *mW*. Table 1 displays the breakdown of total power consumption among the different sections of the chip at this rate. Power consumption in the control section remains a small percentage despite the increased complexity required to implement the memory and datapath power saving strategies. The peak performance at 5.0 *V* generates 60 Mpixels/sec of three RGB components with a 120 *MHz* clock frequency while dissipating 1.2 *W* [5].

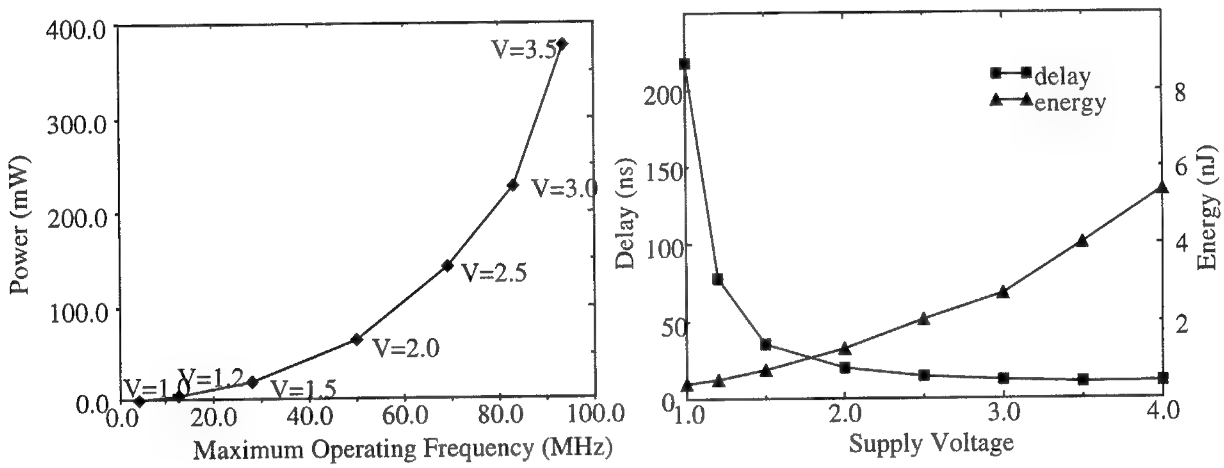


Figure 6. Performance measurements.

Table 1. Measured and estimated power by operations.

Operations.	Mops/sec	Energy/op (pJ)	Estimated Power (mW)	Measured Power (mW)	% of Total
Total Datapath			0.35	**0.34**	**29**
Add (16 bits)	17.7	7	0.12		
3-2 add (16 bits)	25	2	0.05		
Latching (16 bits)	100	1.8	0.18		
Internal Memory			0.26	**0.39**	**33**
Internal Read (16 bits)	2.4	36	0.09		
Internal Write (16 bits)	2.4	71	0.17		
External access (16 bits)	2.7	80	0.22	**0.38**	**31**
Control			0.13	**0.09**	**7**
Total			1.0	**1.2**	**100**

Table 2. Higher resolution configurations.

Format	Size	# chips	Clock freq. (*MHz*)	Power (*mW*)
Sharp LCD	176x240x30	1	3.2 (1V)	1.2
CD-I (TV)	352x240x30	2	3.2 (1V)	2.4
CCIR Video	704x480x30	3	8.4 (1.2V)	12.6
HDTV	1920x1035x75	8	19 (1.5V)	122
Hires Monitor	1024x768x75	4	37 (2.0V)	192

For higher-resolution images, multiple chips would be cascaded, each operating on a maximum of 256 pixels wide slice, producing a final image without boundary artifacts. Table 2 illustrates the power dissipation at the required clock frequency when used for decompressing high-resolution images. The operating voltages are determined by the real-time computation requirements. This parallelism keeps the operating frequency and thus the supply voltage low, resulting in extremely low power dissipation even for HDTV applications. Figure 7 is a chip micrograph of the subband decoder which contains 415,000 transistors in a 9.5 x 8.7 mm^2 area implemented in a 0.8μ CMOS technology.

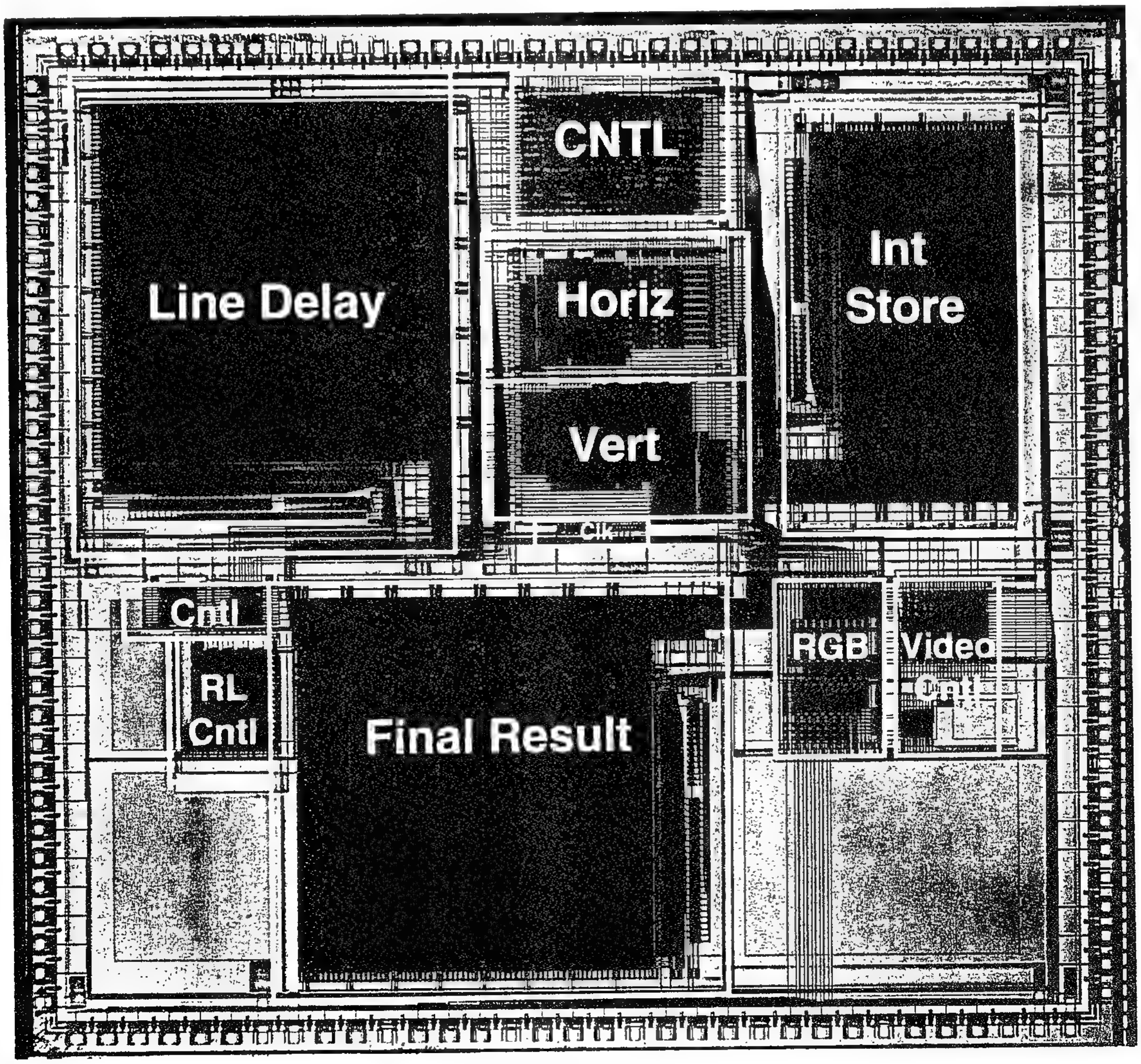


Figure 7. Subband decoder chip micrograph.

2.2. Pyramid VQ Decoder Chip

This section describes the design of a pyramid vector quantization (PVQ) decoder chip used together with the subband decoder chip for real-time video decompression. The chip is designed for low power operation for portable applications, operating at a 1.5 *V* supply and consuming 6.6 *mW* at 6.4 *MHz* clock frequency, sufficient to decode 1.27 Mpixels/sec of color video with 160x240 pixels/frame at 30 frames/sec. The chip integrates 272K transistors on a 9.7mm x 13.8 mm die and was implemented using a 0.8-μ m triple-metal CMOS technology.

2.2.1 Pyramid VQ

Pyramid vector quantization (PVQ) is a fast quantization technique which offers good compression performance (approaches optimal entropy-coded scalar quantization), large reduction in memory requirements compared with standard VQ schemes, and error-resilient properties due to the fixed-length nature of PVQ codewords. Previous work in PVQ has focused on algorithm development [6][7], and an architecture of a PVQ encoder processor was proposed [15].

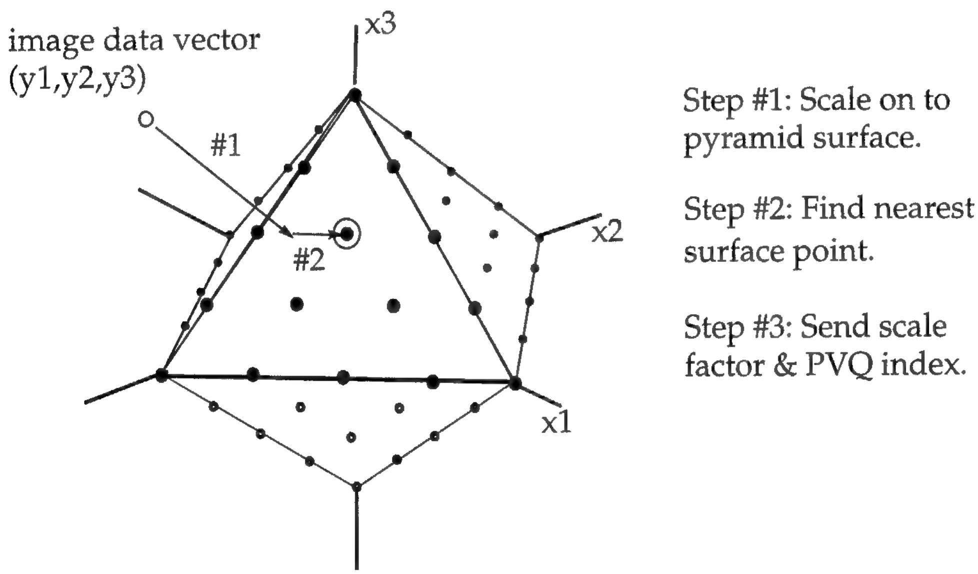


Figure 8. PVQ encoding on a 3-D pyramid surface.

The PVQ encoding and decoding process is as follows: the encoder codes a data vector, formed with L image pixels, by scaling the vector onto an L-dimension pyramid surface and finding the nearest lattice point on the pyramid (Fig. 1). Both the scaling factor and the index corresponding to that lattice point are transmitted and used to decode the quantized vector. Since the lattice points on the pyramid are regularly spaced and can be described by simple recursive equations, encoding and decoding PVQ indices can be done with arithmetic computation, mostly using shifts, subtracts, and compares [12]. For more details on the algorithm, please refer to Appendix A.

2.2.2 PVQ Decoder Architecture

The PVQ decoder performs the following function: it parses an encoded bit stream into PVQ indices and scale factors, decodes the indices into a data vector, scales the vector, and places the results in an output buffer. The overall block diagram in Figure 9 shows the general dataflow of the decoder.

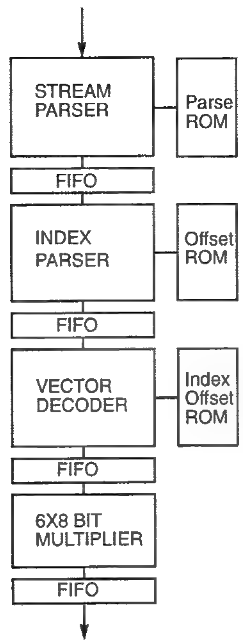


Figure 9. PVQ decoder chip block diagram.

The PVQ decoder is divided into four processing elements. The *stream parser* parses 16-bit input words into PVQ indices and scale factors using a series of two 32-bit shifters. Variable word lengths of indices and scales for various subbands are programmed into a ROM. The *index decoder* (16-bit datapath) decodes each index into four intermediate indices which describe the vector: the *non-zero index* (describes the number of non-zero vector elements), *pattern index* (describes positions of non-zero elements), the *shape index* (describes the values of non-zero elements), and the *sign index* (describes the signs). It decodes the non-zero index by comparing the original PVQ index to offset values stored in a ROM. The remaining three indices are bit concatenated and are simply parsed using shifters. The *vector decoder* (8 to 16-bit datapath) generates a data vector by iteratively comparing the pattern and shape indices to pre-computed offsets stored in a ROM, whose address is determined by the non-zero index. The resulting non-zero values are then negated according to the sign index. Finally, a 6x8-bit pipelined multiplier performs the final scaling of each vector element.

In addition to operating at a low supply voltage, the PVQ decoder chip employs several key architectural strategies to minimize its power consumption and maximize its throughput:

- **Independent Processing Elements and Clocking**

The PVQ decoding algorithm is non-deterministic, i.e., the number of processing steps to decode a PVQ index depends on the vector data. As a result, the latency in the index parser and vector decoder is also non-deterministic. To reduce power consumption and maximize throughput, the four processing elements operate independently, each with its own local control and each individually pipelined. When

idle, each processor enters a power-conserving stand-by mode, and the clock to that unit is gated. Each processor is separated by FIFO buffers and only continues processing when its input FIFO is not empty and its output FIFO is not full. Internal clocking within the FIFO is also turned off when the FIFO is idle. More than half of the chip's total clock capacitance lies in gated clocks. With the exception of the vector decoding unit, whose idle time is typically 10%, the other processing elements are typically idle 50% to 60%. These high idle times lead to savings in total clock power dissipation by roughly a factor of 2.

Finally, true single-phase clocking was selected over two-phase clocking to reduce the significant power consumed in distribution of two clocks.

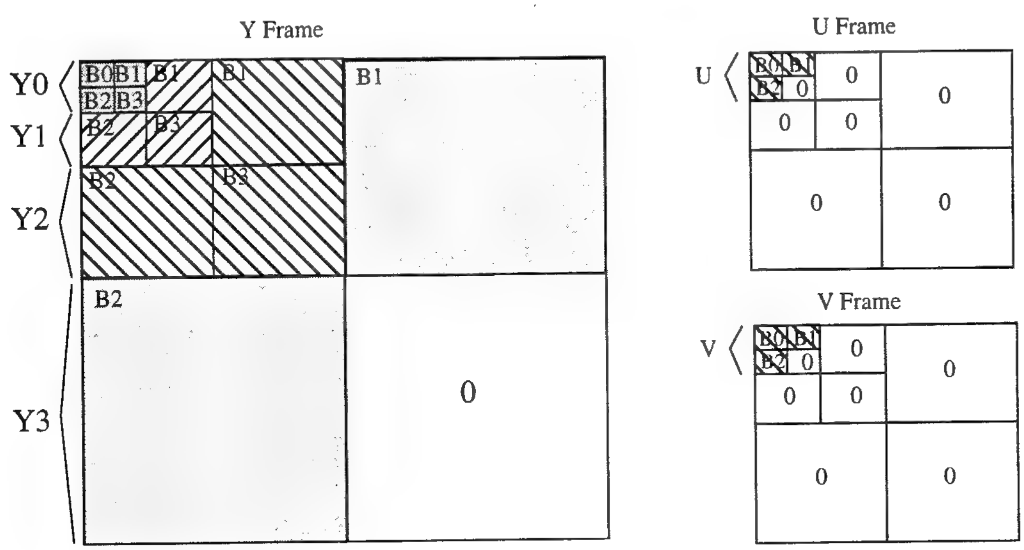
- **Data Interleaving and Elimination of a Frame Buffer**

A major goal of this project is to eliminate the need for external memory for the chip set to maintain low power. To eliminate the uncompressed frame buffer, which is commonly found in hardware decompression decoder systems and very power expensive, the PVQ and subband decoder chips perform line-by-line processing in real time, instead of operating on an entire frame. All required line buffers are on-chip.

In order to process the incoming video frame line-by-line, the data between the different subbands and levels had to be interleaved in the following order:

Y0	Y1	Y2	Y3	U	V	Y3	Y2	Y3	Y3	Y1	Y2	Y3	Y3	Y2	Y3	Y3

where Y0, Y1, Y2, Y3, U, and V are subband levels defined below:



This interleaving scheme resulted in additional control complexity. Each processing unit has its own controller, each keeping track of its location in the interleaving scheme. Despite an increase of about 25% more logic in each controller, we found that this was clearly a good trade-off for reducing the overall system power.

- **Block Search in the Vector Decoding Unit**

For typical image data, the *vector decoder* datapath constitutes the critical path of the chip. Increasing the throughput of this unit directly increases chip throughput, reduces processing cycles, and reduces the amount of output buffering required to meet real-time throughput requirements. Direct implementation of the PVQ algorithm would require a linear search to locate and compute the correct index offset and decode the vector. Improved throughput was achieved by searching and processing a block of 4 vectors at a time. For typical image data, this reduced the average number of search iterations from 15 to 3, and halved the number of processing cycles and the amount of output buffering.

- **External Accesses and Run-Length Encoding**

All intermediate results are stored on-chip in FIFO buffers. All stream parsing data and pre-computed factorial and combinatorial offset values, required for index parsing and decoding, are stored in on-chip ROM's. This approach eliminated the need for external accesses to off-chip memory, which are power expensive.

Data vectors that contain a large numbers of zeros, as found in high spatial frequency image data, are run-length encoded to compress the representation of consecutive zeros. This encoding reduces the amount of external accesses by a factor of 3, output buffering by a factor of 3, and the number of internal buffer accesses by up to a factor of 10. Here, we traded off additional control to perform zero run-length encoding for lower I/O power and on-chip data buffering.

2.2.3. Power Consumption

Given the non-deterministic nature of the decoding algorithm, we simulated (using a switch-level simulator) the energy consumed by the PVQ decoder chip for an average case set of input data. The vectors used to derive the data shown in Table 3 represent a typical sample set of the PVQ codes collected from a wide range of images, based on a 1.27 Mpixels/sec (176×240 pixels/frame at 30 frames/sec) rate to match that of the subband decoder. Note that the PVQ decoder must run at twice the clock frequency as the subband decoder to meet the same throughput.

The PVQ decoder chip consumes roughly twice the power of the subband decoder chip. The doubled clock frequency of the PVQ decoder explains most of this difference. A third of the chip power is consumed by the register FIFO storage. Further analysis shows that about 55% of this FIFO power is consumed by local clock lines. It is also interesting to note that the datapath power is roughly equal to the control power. There are several reasons for this: (1) the datapaths use local gated clocks, while the control sections do not -- a significant factor in that the chip may be idle up to 40% of the time, and (2) the control sections require greater complexity due to the non-deterministic nature of the algorithm and the data reordering scheme. Because of the relative large design (1 *cm* x 1.4 *cm*), extensive global clock routing results in significant power consumption by the clock lines. Overall the total power consumed on clocking, including global clocking and local clocking in the control, datapaths, and FIFOS, makes up about 50% of the chip power. Finally, the power consumed in external accesses is a relatively small fraction, because the design limits off-chip accesses to only input compressed data and output run-length encoded data.

Figure 10 shows the actual power performance of the chip for various clock frequencies. At 6 *MHz*, the target rate for real-time video at 30 frames/sec, the measured power is 6.4 *mW*. The additional 1.8 *mW* is accounted for by an error on the chip which caused additional DC power in a ROM.

Table 3: Simulated power breakdown of the PVQ decoder at a clock frequency of 6.4 *MHz*.

Operations	**Ops/pixel**	**Mops/sec**	**Energy/op (pJ)**	**Power (mW)**	**%**
Total datapath:				0.83	17.9%
CPQ (16 bits)	2.3	2.9	16.1	0.05	
Shifts (16 bits)	1.1	1.4	8.3	0.01	
Multiply (8 x 6 bits)	0.7	0.9	26.0	0.02	
Registers (16 bits) (including clock)	38.7	49.1	15.3	0.75	
Control	0.4	0.53	177.1	0.93	20.0%
FIFOS					
Local clock	4.7	6	153.3	0.92	19.9%
Registers/control	39	44.7	15.3	0.72	15.6%
Global clock	4.7	6.0	107.4	0.64	13.8%
Global half clock	2.4	3.0	21.3	0.06	1.3%
Clock buffers	3.2	4.1	44.3	0.18	3.9%
ROM access (16 bits)	1.5	1.9	98.0	0.19	4.1%
External access (16 bits)	0.6	0.7	180.0	0.16	3.5%
Total				**4.63 mW**	**100%**

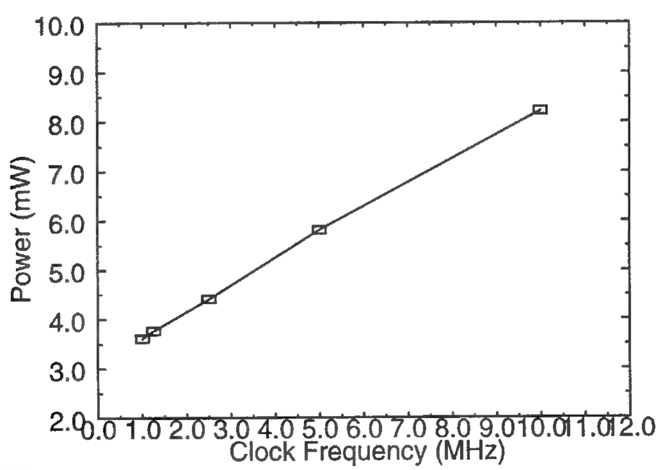


Figure 10. Measured power vs. clock frequency at 1.5 *V* operation.

Table 4. The PVQ chip summary.

Technology	0.8-µm N-well CMOS
Power	6.4 *mW* @ 1.5 *V* and 6.4 *MHz*
Die Size	9.7 mm x 13.8 mm
Transistor Count	272 K

2.3. Portable Video-on-Demand System Prototype

Figure 11 shows the photo of our portable decoder module, with the subband decoder, PVQ decoder, and D/A converter marked. This board occupies an area of 3"x 5", with several feedback op-amps to control the analog signals to the color display.

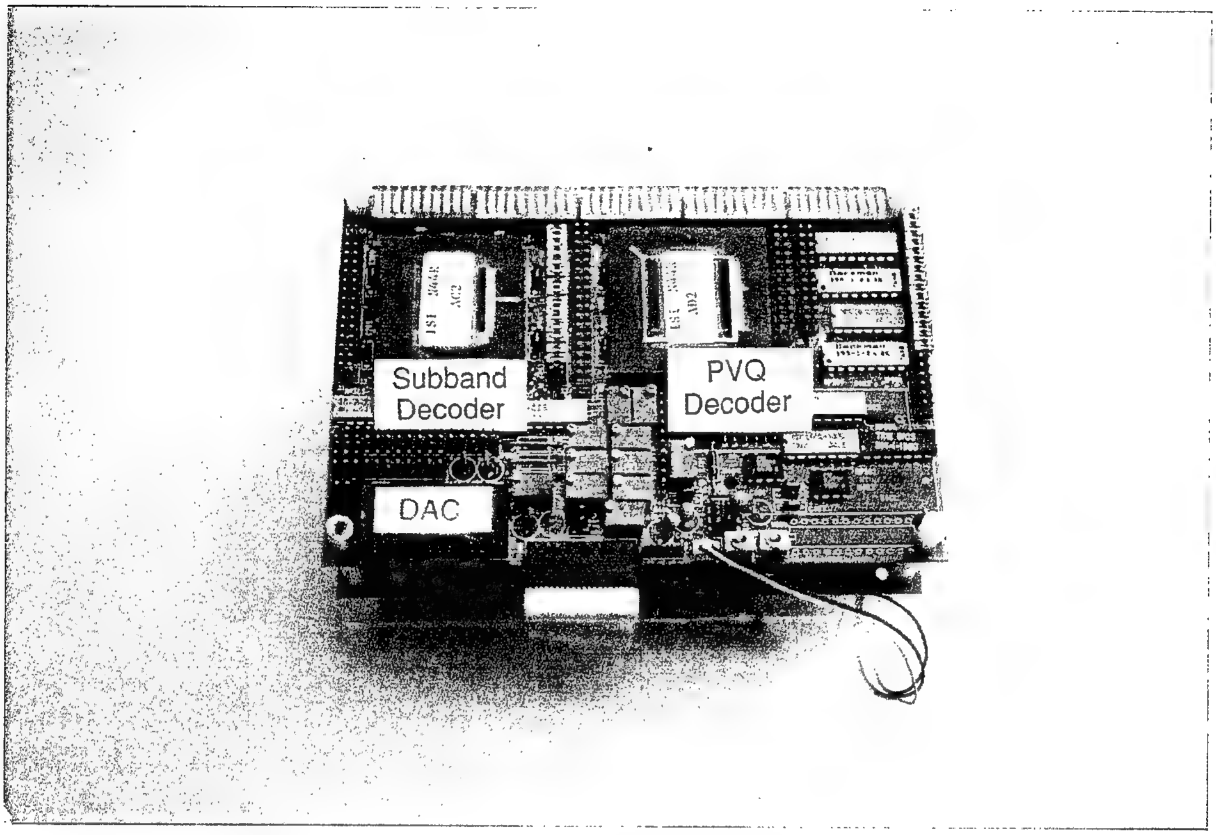


Figure 11. The portable decoder prototype.

The complete video-on-demand system consists of a portable decoder with a color LCD display that decodes and displays compressed video sequences, a radio transmitter and receiver, and an encoding base station implemented in a DSP multiprocessor board. The wireless data transmission is provided by three pairs of direct-sequence spread spectrum radio transceivers manufactured by Proxim, delivering a raw data rate at 727 *Kbits/sec*. The decoding chip set on the portable decoder receives compressed video date and decompresses them to RGB color components, which are then converted to analog signals for the color display. The display is a 4" color thin film transistor active matrix display with a resolution of 160 pixels by 234 lines.

As shown in Figure 12, the video-on-demand system accepts video data from two sources, a video server with a compressed video database and an NTSC camera. Video sequences stored in the video database are pre-compressed using our PVQ/subband encoding algorithm. As the compressed bit rate ranges from 0.5 Mbits/sec to 1 Mbits/sec, a simple bus interface between the video server (a SUN

workstation) and the radio transmitter has been built to support this constant rate of data transfer. Framing information necessary for synchronization at the decoder end is also added at this interface.

System Block Diagram

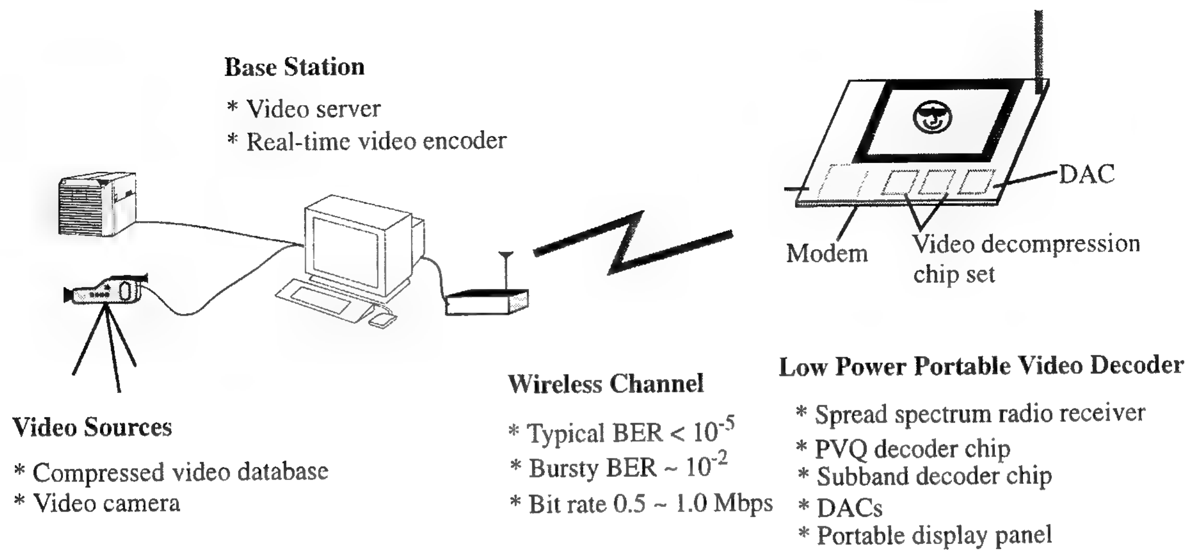


Figure 12. The portable video-on-demand system.

The video-on-demand prototype system includes a real-time encoder to allow for live video sources such as a camcorder as an input device. The encoder consists of an NTSC decoder and multiple TMSC40s on a DSP multiprocessor board, which implements the PVQ/subband encoding algorithm in real time. As our PVQ/subband algorithm is a symmetric compression algorithm, implying that the encoding and decoding procedures are almost identical, a low-power encoder would be feasible if a portable system of two-way video communication is desired.

Figure 13 shows the photo of the portable video-on-demand system prototype, which comprises a camera and a VCR, a real-time encoder box, and the portable decoder module. For wireless transmission, a radio transceiver board is embedded in the encoder box. The portable decoder module consists of an LCD display and two PC boards, one of which is for radios and I/O control and the other is the real-time video decoder board as shown in Figure 11.

From designing this prototype video-on-demand system, we learned that power reduction can be best attained through algorithm and architecture decisions, guided by the knowledge of underlying hardware and circuit properties. This hardware-driven algorithm design strategy is key to delivering high-quality video at an extremely low power level.

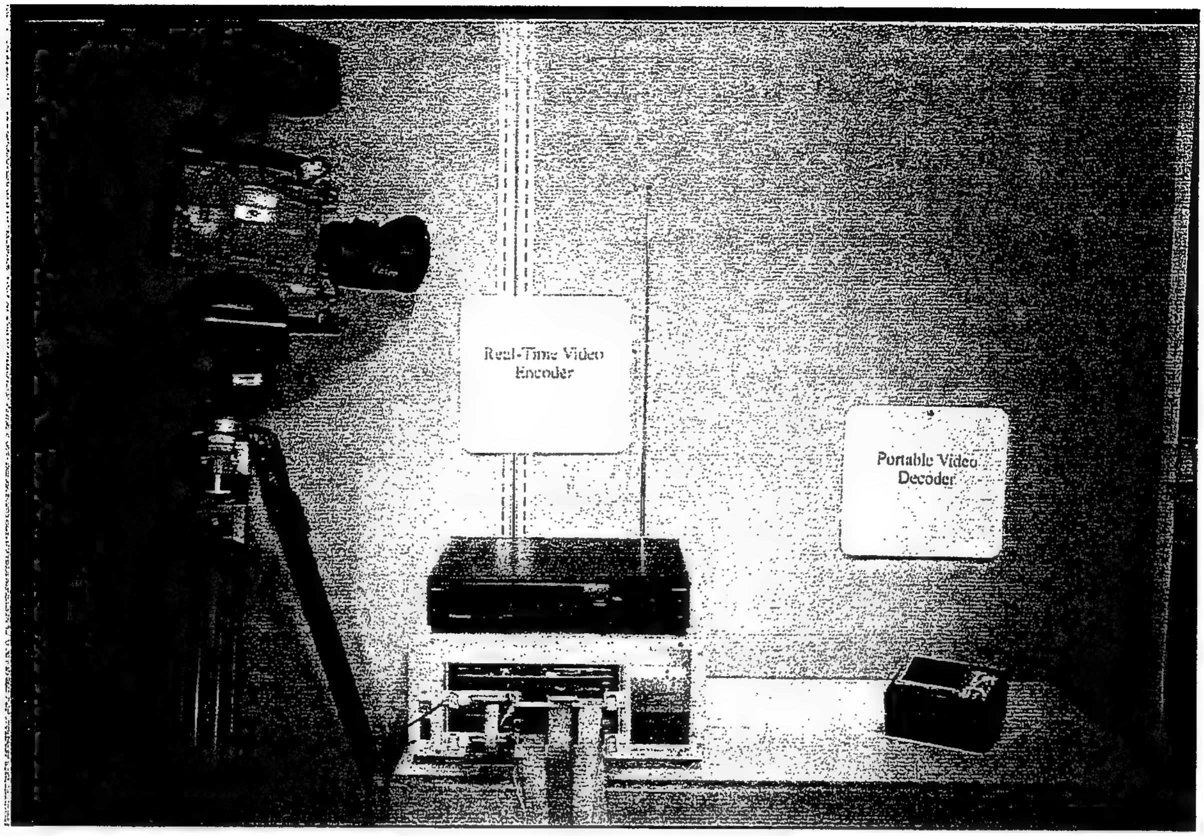


Figure 13. The portable video-on-demand system prototype.

3.0 Perception-Based Scalable Compression

This section presents an algorithm for scalable compression using tree structured vector quantization (TSVQ) of perceptually weighted generic block, lapped, or wavelet transforms. The algorithm produces an embedded bit-stream to support decoders with various spatial and temporal resolutions. Bandwidth scalability with a dynamic range from a few Kbps to several Mbps is provided. The algorithm further supports decoders with varying alphabet size, computation, memory, latency, and power requirements. The embedded bit-stream produced is prioritized with bits arranged in order of visual importance. The algorithm also allows easy joint-source channel coding on heterogenous networks. The subjective quality of compressed images improves significantly by the use of perceptual distortion measures.

A typical application of scalable compression is multicast over heterogenous networks consisting of ATM, Internet, ISDN, and wireless networks having various bandwidths, and hosting decoders with various spatial and temporal resolutions, reproduction alphabets, limits on computation, memory, power, and latency, etc. Scalable compression is also important in image browsing, multimedia applications, transcoding to different formats, and embedded television standards. It can also be used to overcome congestion due to contention of network bandwidth, CPU cycles etc., in the dynamic environment of general purpose computing systems.

Most popular existing compression algorithms do not have the desired properties of scalable compression. Compression standards like MPEG-2 offer scalability to a limited extent and lack the dynamic range of bandwidth. Recently there has been some work on scalable compression by Taubman and Zakhor [16] and Shapiro [17]. Taubman and Zakhor [16] use 3-D subband coding with layered progressive scalar quantization and adaptive arithmetic coding. Shapiro uses zero-tree scalar quantization of wavelet coefficients with adaptive arithmetic coding.

Our work differs in several respects. We achieve scalable compression using arbitrary block transforms (e.g., discrete cosine transform, Walsh-Hadamard transform, Haar transform), lapped orthogonal transforms (LOTs), or wavelet transforms; we use tree-structured vector quantization [10] instead of scalar quantization as the quantization scheme; and we use a perceptually based input-weighted squared error distortion measure in the design of the TSVQ to achieve better visual quality of the compressed images.

3.1 Problem Statement

Let θ be a vector of decoder characteristics, consisting of such components as bit rate in bits per second, display size in pixels, display depth in bits, frame rate, etc. Let Λ be the space of such characteristics, so that $\theta \in \Lambda$. For a given source of data, let α_θ and β_θ denote the optimal encoder and decoder with characteristic vector θ, and let π_θ denote the corresponding optimal average performance (in distortion and computational complexity, say). Note that for a fixed input signal X, the bit streams $\alpha_\theta(X)$ for different $\theta \in \Lambda$ are different in general. Now let α be a single encoder, and for each $\theta \in \Lambda$, let β'_θ be a decoder with characteristic vector θ, such that the average performance of α and β'_θ is π'_θ. We say that the data compression system α, β'_θ, $\theta \in \Lambda$, is *scalable* with respect to Λ if for every $\theta \in \Lambda$, π'_θ is close to π_θ. This definition captures the intuitive notion that a data compression system is scalable if for any given signal the encoder produces a single bit-stream that can be decoded by a variety of decoders with different characteristics. However it rules out as scalable trivial systems such as one in which each decoder ignores the bit-stream and reproduces blank images in the right format, or systems in which each decoder decompresses the bit-stream into an intermediate image, say $\tilde{X} = \beta(\alpha(X))$, and then recompresses the intermediate image to produce an output image in the right format, say $\hat{X} = \beta_\theta(\alpha_\theta(\tilde{X}))$. In each of these cases, π'_θ would not be close to π_θ, in either distortion or computational complexity.

We now list a number of properties, including scalability, that our compression algorithm achieves.

- **Rate or bandwidth scalability:** The algorithm supports a range of bandwidths (e.g., 10 Kbps - 10 Mbps).
- **Resolution scalability:** The algorithm supports decoders with different display sizes (e.g., 640x480, 320x240, 160x120).
- **Alphabet scalability:** The algorithm supports decoders with different alphabet sizes (1- to 8-bit gray, or 2- to 24-bit color).
- **Computation scalability:** The algorithm supports decoders with different computation capabilities (e.g., PCs, workstations, special-purpose hardware, parallel computers).
- **Power scalability:** The algorithm supports decoders with different power limitations.
- **Memory scalability:** The algorithm supports decoders with different amounts of memory.

- **Latency scalability:** The algorithm supports decoders with different latency requirements.
- **Joint-source channel coding:** The algorithm is compatible with channel-error concealment techniques.
- **Perceptual Coding:** The algorithm takes into account human visual perception when performing compression.

3.2 Tree-Structured Vector Quantization with Perceptual Distortion Measures

Embedded coding and vector quantization are combined to achieve the above mentioned goals. Both embedded coding and vector quantization can be performed by tree-structured vector quantization (TSVQ). TSVQ can be thought of successive approximation of ordinary vector quantization (VQ) [10]. In ordinary VQ, the codewords lie in an unstructured codebook, and each input vector is mapped to the minimum distortion codeword. This induces a partition of the input space into Voronoi encoding regions. In TSVQ, on the other hand, the codewords are arranged in a tree structure, and each input vector is successively mapped (from the root node) to the minimum distortion leaf node. This induces a hierarchical partition, or refinement of the input space as the depth of the tree increases. Because of this successive refinement, an input vector mapping to a leaf node can be represented with high precision by the path map from the root to the leaf, or with lower precision by any prefix of the path. Thus TSVQ encodes the input to an embedded code. If the depth of the tree is R and the vector dimension is k, then bit rates $0/k, 1/k, \ldots\ldots, R/k$ can all be achieved. Algorithms for designing TSVQs and its variants have been studied extensively. For example, variable-rate TSVQs can be constructed by varying the depth of the tree. This can be realized by "greedily growing" the tree one node at a time [20], or growing a large tree and pruning it back subject to a constraint on its average length or entropy [19].

Instead of using the mean squared error as our distortion measure we use subjectively meaningful distortion measures in the design and operation of our TSVQ. For this purpose we can transform pixel vectors using either a generic block transform, lapped orthogonal transform, or discrete wavelet transform, and then apply the following input-weighted squared error to the transform coefficients:

$$d_T(y, \hat{y}) = \sum_{j=1}^{K} w_j \left(y_j - \hat{y}_j\right)^2$$

where y_j and $\hat{y}_j$ are the components of the transformed vector y and the corresponding decoded vector $\hat{y}$, and w_j is a component of the weight vector depending on y. That is, the distortion is the weighted sum of squared differences between the coefficients of the original transformed (block, lapped, or wavelet transform) vector and the corresponding decoded vector. The vector y is constructed by selecting either the full block of transform coefficients in the case of block or lapped transforms or transform coefficients across spatially localized bands in the case of wavelet transforms.

The weight vector reflects human visual sensitivity to quantization errors in different transform coefficients. When used in the perceptual distortion measure for vector quantization, the weight vector controls an effective stepsize, or bit allocation, for each band. When the transform coefficients are vector quantized with respect to a weighted squared error distortion measure, the weights $w_1, \ldots, w_K$ play a role similar to the stepsizes in the scalar quantization case. By incorporating the perceptual model into the TSVQ distortion measure, rather than into a stepsize or bit allocation algorithm, the weights can vary with

the input data characteristics and the decoder can still operate without the encoder transmitting any side information on what the weight vector is.

3.3 Generic Block Transform based Scalability

In the first stage of our encoder (Figure 14) an image is transformed using a generic block transform (e.g. DCT, WHT, HT, LOT). The second stage of the encoder forms a vector of the transformed block. Next the block transform coefficients are vector quantized using a TSVQ designed with a perceptually meaningful distortion measure. The encoder sends the indices as an embedded stream with different index planes. The first index plane contains the index for the rate- $1/k$ TSVQ codebook. The second index plane contains the additional index which along with the first index plane gives the index for the rate-$2/k$ TSVQ codebook. The remaining index planes similarly have part of the indices for TSVQ codebooks of rates $3/k$, $4/k$, ..., R/k respectively. The advantage of this hierarchical encoding is that it produces an embedded prioritized bit-stream. Thus rate or bandwidth scalability is easily achieved by dropping index planes from the embedded bit-stream. The decoder can use the remaining embedded stream to index a TSVQ codebook of the corresponding rate.

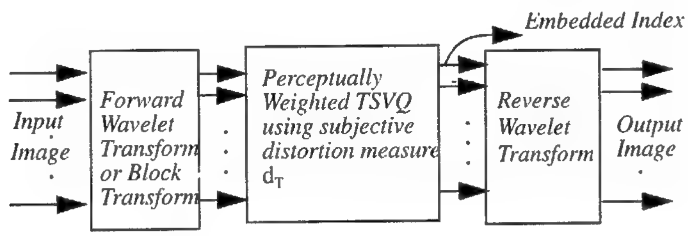


Figure 14. Block diagram of the scalable encoder.

The decoder uses a look-up table to store the TSVQ decoder codebook of a specific rate which stores pre-computed inverse block transforms already performed on each codeword, eliminating the need to calculate inverse block transforms in real time. For achieving resolution scalability the decoder can store codebooks with different sizes of vectors obtained by decimation of the original full-resolution decoder codebook or performing non-linear interpolative VQ [18]. To achieve alphabet scalability we can color quantize the resulting codewords of the decoder codebook using the color quantization algorithm proposed by Chaddha et. al. [21], which was described in a previous semi-annual report in 1993.

Tree-structured vector quantizers achieve computation scalability proportional to bandwidth as the computation required in table lookups is proportional to the depth of the index planes used. By using a hierarchical memory scheme the decoding algorithm provides power scalability as well. The algorithm also provides memory scalability since the memory requirements of the decoder is determined by the number of index planes it can store, the display resolution and the frame rate. Our algorithm currently has no frame latency since it does not perform any temporal operations. In future work temporal compression and latency scalability will be provided. The objective again will be to map the latency requirements of the decoder into bandwidth requirements. The scalable compression algorithm also allows easy joint source-channel coding since the routers/switches can react to congestion by intelligently dropping index planes from the embedded bit-stream. Next we describe wavelet transform based scalability.

3.4 Wavelet Transform based Scalability

In the first stage of our encoder (Figure 14) an image is run through separable horizontal and vertical wavelet filters such as described in Figure 2, creating frequency subbands at different levels.

The second stage of the encoder forms a vector across the different subbands of the wavelet transformed image, which is vector quantized using TSVQ designed with a perceptually meaningful distortion measure. The encoder sends the indices as an embedded stream in the same way as for the block transform case. Thus scalability is achieved in the same way as for the block transform case.

The decoder looks up the reproduced wavelet transform vector from the TSVQ codebook of a specific rate. For resolution scalability the decoder codebook contains only the subband coefficients of the required size. For wavelet synthesis, the lowest level subbands are each upsampled and filtered, then summed together to form a reconstructed subband. This process is repeated through all levels until the final image of the required size is formed/

3.5 Simulation Results

Table 5 gives the PSNR (peak SNR) results on the 8-bit monochrome image Lena (512x512) at different compression ratios encoded using three algorithms: plain TSVQ (unweighted), DCT followed by perceptually weighted TSVQ (DCT+TSVQ) and wavelet transform followed by perceptually weighted TSVQ (Wavelet+TSVQ). The codebooks for the different forms of TSVQ for these simulations have been generated by training on 5 different images (Woman1, Woman2, Man, Couple and Crowd).

Even though the PSNR numbers are lower for perceptually weighted TSVQs, the subjective quality of the images compressed using perceptual weighting is significantly better than the unweighted TSVQ's at each compression ratio [26]. By incorporating perception into scalable compression techniques, we provide a compression framework in which images of the best possible quality can be delivered given any available resources.

Table 5. PSNR results of scalable compression algorithms.

Compression Ratio	TSVQ	DCT + TSVQ	Wavelet + TSVQ
8	36 dB	35.4 dB	35.5 dB
12	34.3 dB	33.7 dB	33.9 dB
14	33.1 dB	32.5 dB	32.6 dB
18	31.6 dB	31.2 dB	31.4 dB
28	30.1 dB	29.6 dB	29.9 dB
47	29.2 dB	28.7 dB	28.9 dB
67	28.3 dB	27.9 dB	28.2 dB
90	27.4 dB	27 dB	27.2 dB
130	26 dB	25.7 dB	25.8 dB

4.0 An End-to-End Software Only Scalable Video Delivery System

Pre-compressed video delivery systems commonly operate at fixed data rates even though scarcity of network bandwidth and processor cycles is a common occurrence in the dynamic environment of general-purpose computing systems. Scarcity arises from the outright lack of resources (e.g. network bandwidth or CP cycles), contention for available resources due to congestion, or a user's unwillingness to allocate

needed resources to the task. A scalable video delivery system has greater flexibility and therefore can more effectively deliver video in the presence of system resource scarcity. This section describes an end-to-end system combining a scalable video compression algorithm, video delivery software, a software video decoder, and a market-based mechanism for the resolution of conflicts in providing video to the user. In contrast with existing schemes, this approach provides clients with the ability to trade off video quality for system resources, permitting a much higher degree of overall value to be delivered with a given configuration of hardware.

Sun Microsystems Laboratories and Stanford University are building a number of applications and services which require video storage, processing, and transmission as a component. The two primary services are a video library and an interactive lecture distribution system. A hierarchical video storage systems is being built at Sun Microsystems Laboratories.

The video encoding process creates an embedded video stream from which different streams at different resolutions (both spatial and temporal), and different rates, can be extracted depending on the capabilities and requirements of the decoders. In this system, the decoding subsystem defines the spatial and temporal resolutions of its displayed video stream (i.e., either 160x120, 320x240, or 640x480 pixels per frame, and from 1 to 30 frames per seconds). The various video quality specifications result in communication bandwidth scalability with a dynamic range from 10 Kbps to 10 Mbps. A low-cost, software-based decoder of the scalable video stream has been developed which primarily uses table lookups and additions to decode frames of video. A disk-based video server has also been implemented which makes use of careful layout and scheduling to support multiple clients of the prerecorded video streams. In addition, the system provides support for media synchronization and makes use of an electronic-market-based mechanism to provide a complete solution for scalable end-to-end video delivery.

Most popular existing compression systems do not have the desired properties of scalable compression. Compression standards like MPEG-2 offer scalability to a limited extent and lack the dynamic range of bandwidth. Our work differs in that it provides an end-to-end software only solution. The software-based video decoder should use minimal CPU resources on a range of systems. Inside the network, it must be possible to scale the embedded video stream to fit into a lower bandwidth link or to adapt to congestion. In addition, there should be error resilience in the decoder algorithm to allow for communication errors, such as bit errors or cell loss. The end to end system also requires support for audio-video synchronization and a scalable, multiple-user video storage system. Finally, there should be a simple mechanism to transform the user's selection of a delivery bandwidth to choose the most appropriate point in the spatial resolution, temporal resolution, data-rate and quality space.

4.1 Scalable Compression

To meet the goal of a low-cost, scalable video delivery system, we used a scalable compression algorithm similar to that described in Section 3, which produces an embedded bit-stream that can easily be rescaled by dropping less important bits from the video stream. Then a low-cost, software-based decoder of the scalable video stream is developed which only performs table lookups and additions to decode a frame of video. The disk server utilizes the embedded stream of video to scale to the appropriate network bandwidth. Finally, the system utilizes an existing media synchronization framework to provide a complete solution for end-to-end video delivery.

4.1.1 Encoding Structure

The video coding algorithm is based on a Laplacian pyramid decomposition [22] (see Figure 15). The original 640x480 image is decimated to 320x240 and 160x120 pixels for encoding. The base image of 160x120 pixels is compressed and then decompressed. The resulting decompressed image is upsampled and subtracted from the 320x240-pixel image to give an error image. The error image is compressed and transmitted. The decompressed 160x120-pixel image is also upsampled to 640x480 pixels. Then it is subtracted from the original 640x480-pixel image to give an error image which is then compressed. Thus the encoding stage consists of three resolutions. The base layer transmitted has the compressed data for 160x120-pixel images. The enhancement layer has the error data for the 320x240-pixel and 640x480-pixel images.

The decoder can support up to three spatial resolutions, i.e. 160x120, 320x240 and 640x480 pixels. It can further support any frame rate as the frames are coded independently.

In order to achieve bandwidth scalability with an embedded bit-stream we use the tree-structured vector quantization (TSVQ) as described in Section 3.

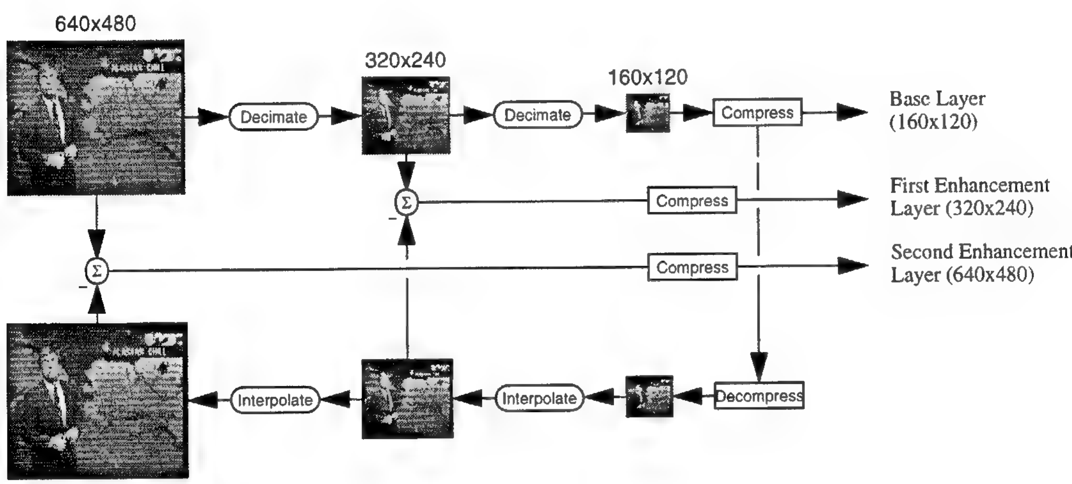


Figure 15. Block diagram of the Laplacian pyramid encoding algorithm.

To explain the encoding/decoding process, we use DCT followed by perceptually-weighted TSVQ as an example. The decimated 160x120-pixel image is transformed using DCT. Next the DCT coefficients are vector quantized using a TSVQ designed with a perceptually meaningful distortion measure. The encoder sends the indices as an embedded stream with different index planes. The first index plane contains the index for the rate $1/k$ TSVQ codebook, the second index plane contains the additional index which along with the first index plane gives the index for the rate $2/k$ TSVQ codebook, etc. The advantage of this encoding of the indices is that it produces an embedded prioritized bit-stream. Thus rate or bandwidth scalability is easily achieved by dropping index planes from the embedded bit-stream. The error images for higher resolutions are encoded in the same way.

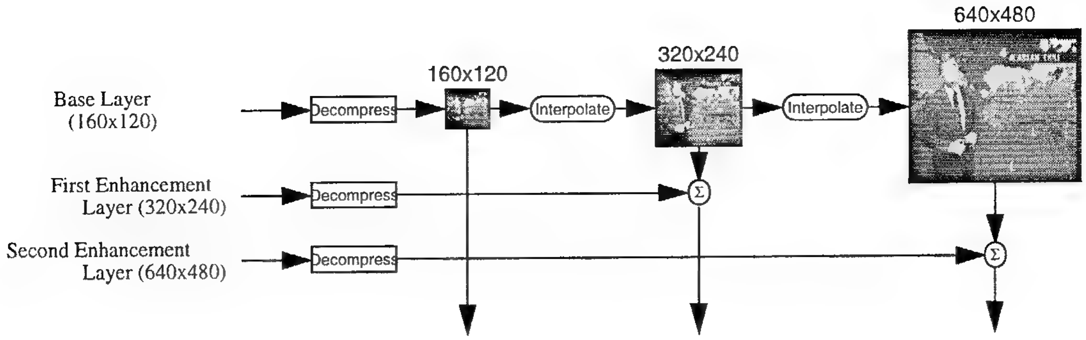


Figure 16. Block diagram of the Laplacian pyramid decoding algorithm.

Frame-rate scalability can be easily achieved by dropping frames as there is no inter-frame compression in the algorithm right now. There is no motion estimation or conditional replenishment in the current work. For future work these schemes will be incorporated in the system.

4.1.2 Decoder Structure

The decoder of our video system is shown in Figure 16. The decoder uses the indices from the embedded bit-stream to lookup from a codebook which uses the processor cache efficiently. The decoding process consists of loading the codebooks into the processor cache and performing table-lookups from it. The base layer is obtained by performing lookups while the enhancement layers are obtained by performing lookups of the base and error images followed by addition.

The TSVQ decoder codebook has the inverse DCT performed on the codewords of encoder codebook. Thus at the decoder there is no need for performing inverse block transforms. Color conversion i.e. YUV to RGB conversion is also performed as a pre-processing step by storing the corresponding color-converted codebook. This is achieved by forming an RGB color vector from the codewords of the codebook and color quantizing them to the required alphabet size. Thus the same embedded bit-stream can be used for displaying images on different alphabet decoders with the different alphabet sizes.

4.2 Disk Layout

The disk layout stores compressed video in two data streams: base layer and enhancement layer streams. The base layer data is stored with the following hierarchy:

Frames: Data for each frame is stored together. Each frame has a set of index planes corresponding to different number of bits used for the lookup.

Scalable Stream: The compressed stream consists of lookup indices with different number of bits depending on the bandwidth and quality requirement. The lookup indices for each frame are stored as groups of index planes pre-formatted with application level headers for network transmission. The 4 most significant bits of the lookup indices are stored together as the first section of the frame block. Then 4 additional 1-bit planes of lookup are stored in sequence as separate sections of the frame block to provide

lookup indices varying from 4, 5, 6, 7, 8 bits. The different lookup indices provide data streams with different bandwidth requirements.

The error data is placed similarly as another data stream. The 2 most significant bits of the lookup indices are stored in the first section for each frame block, then again the next 2 bits of the lookup indices as the second section. Then 4 additional 1-bit planes of lookup indices are stored to provide lookup indices varying from 2, 4, 5, 6, 7, 8 bits.

The video server uses RAID-like techniques [23] to stripe each data stream across several drives. The design allows for recovery from failure of any single disk without diminishing the capacity of the server. Because of the RAID approach, there is no restriction on the number of active users of a given title, as long as they can be accommodated within the servers's total bandwidth. That is, the usage can range from all active users receiving the same stream at different offsets to all receiving different streams.

The streams of base and enhancement layer data are striped in fixed size units across the set of drives in the RAID group with parity placed on an additional drive. The selection of the parity drive is fixed since data updates are extremely rare compared to the number of times the streams are read. The current striping policy keeps all of the lookup indices for an individual frame together on one disk; while this costs some loss of storage capacity due to fragmentation, this policy allows for ease of positioning when a user is single stepping or fast forwarding the display. Use of parity on the stripe level allows for quick recovery after a drive failure at the cost of having substantially more buffer space available to hold the recovery data set.

4.3 Network Layer

The video server utilizes the planar bit stream format directly as the basis of the packet stream in the network transport layer. The embedded stream bits plus the application packet header are read from the disk and transmitted on the network in exactly the same format. For example, the base video layer has the 4 most significant bits of the lookup indices stored together so those bits are transmitted as one 2440-byte packet and each additional index bit plane of the less significant bits is transmitted as a separate 640-byte packet. The header contains a frame sequence number, nominal frame rate, size, a virtual time stamp, and a bit-plane type specifier sufficient to make each packet an identifiable stand-alone unit. The server uses the self identifying header to extract each bit-plane packet from the striped frame data retrieved from the disk subsystem.

The server also uses the time stamp and rate information in the header as the means to pace the network transmission and disk read requests. The server uses a feedback loop to measure the processing and delivery time delays of the disk reads and queue the network packets for transmission. The server then uses these measures to schedule the next disk read and packet transmission activities to match the video stream's frame rate. The server can moderate the transmission rate based on slow down/speed up feedback from the decoder.

The video decoder is responsible for the reassembly of the lookup indices from the packets received from the network. In the event of the loss of one of the less significant bit-plane packets, the decoder uses the more significant bits to construct a shorter lookup-table index yielding a lower quality but still recognizable image.

The use of separately identified packets containing index bit planes makes it possible for networks to easily scale the video as a side effect of dropping less important packets. In networks providing QOS qualifiers such as ATM, multiple circuits can be used to indicate the order in which packets should be dropped (i.e. the least significant bit-plane packets first). In an IP router environment, packet filters can be constructed to appropriately discard less important packets. For prioritized networks the base layer will be sent on the high priority channel while the enhancement layer will be sent on the low priority channel. In order to provide error resiliency the use of a fixed-rate coding scheme with some added redundancy, allows robustness in the face of packet loss. The server supports two usage scenarios:

Point-to-Point demand: In this case each destination system decoder comes with its specific requirements to the server. The server then sends the selected elements of the embedded stream across the network to the destination. A separate network stream per destination allows the user to have VCR style functionality such as play/stop/rewind and fast forward/fast reverse. If congestion occurs on the network, then the routers and switches can drop packets from the embedded stream to give a lesser number of lookup bits.

Multicast: In this case the server puts out the entire embedded stream for different resolutions and rates onto the network as a set of unicast trees. The server has no idea about the decoders at the destinations. There may be one to eleven unicast trees depending on the granularity of traffic control desired. The primary traffic management is performed during the construction of the unicast trees, by deleting the branches of the unicast trees carrying the less important bit streams from the lower bandwidth networks. The network in this case takes care of bandwidth mismatches by not forwarding packets to the networks which are not subscribed to the unicast tree. Switches and routers can still react to temporary congestion by dropping packets from the embedded stream to deliver fewer bits of lookup indices.

4.4 Audio Subsystem

The delivery system treats the audio track as a separate stream which is stored on disk and transmitted across the network as a separate entity. The audio format supports multiple data formats from 8 *KHz* telephony quality (8 bit μ-law) to 48 *KHz* stereo quality audio (2 channel, 16-bit linear samples). Most of the video clips on the current system have 8 *KHz* telephony audio since the intent is to be able to distribute the material over medium to low bandwidth networks. The server has the capability to store separate high and low quality audio tracks and to transmit the quality audio track selected by the user. Since the audio transits the network on a separate circuit the audio can easily be given a higher QOS than the video streams. Rather than loading the networks with duplicate audio packets, we ramp the audio down to silence when packets are lost or overly delayed.

Since audio and video are delivered via independent mechanisms to the decoding system, the two streams must be synchronized for final presentation to the user. At the decoder, the receiving threads communicate through the use of a shared memory region, into which the sequence information of the current audio and video display units are written. Since the human perceptual system is more sensitive to audio dropouts, the decoder uses the audio codec as the master clock for synchronization purposes. As the streams progress, the decoder threads post the current data items' sequence information onto the "blackboard", and the slave threads (such as the video decoder) use the posted sequence information of the audio stream to determine when their data element should be displayed. The "slave" threads then delay

until the appropriate time if the "slave" is early (more than 80 milliseconds ahead of the audio). If the "slave" data is too late (more than 20 milliseconds behind the audio), it is discarded on the assumption that continuing to process late data will delay more timely data. The video decoder can optionally measure the deviation from the desired data delay rate and send speed-up and slow-down indications back to the video server. This process synchronizes streams whose elements arrive in a timely fashion and does not allow a slow stream to impede the progress of the other streams.

4.5 Costing Structure

In the event of scarcity of resources, some global prioritization of user requests must take place or overload collapse is likely. This system utilizes payment for services and resources as the means of defining the overall value of each resource allocation decision. Given these values, a total ordering of the user requests can be made and the less important requests can be dropped. The user specifies what he or she is willing to pay for a given service; this proposed payment along with the required resources (network and disk bandwidth) are submitted to an electronic market which uses micro-economic models to decide what amount of bandwidth resource is available to the user [24]. For that particular bandwidth a table is looked at which gives the best possible combination of spatial resolution, frame rate and the number of index planes to give the best quality of decompressed video. This table is built using a subjective distortion measure [25]. The user also has an additional option of specifying the spatial resolution, frame rate and bandwidth directly.

4.6 Scalable Video System

The overall system combines the compression algorithm, disk management, network transport, decoder and synchronization mechanisms to provide an end-to-end scalable video delivery service. The service is divided into three groups of components: preprocessing, media server, and media player.

The preprocessing components are audio capture, video capture, video compression, and a data stripping tool. The video is captured and digitized using single step VCR devices. Then each frame is compressed off-line using the encoding algorithm. Currently, it takes about one second to compress a frame of video data and the single step VCR devices can step at a one frame per second rate so capture and compression can be overlapped. The audio data is captured as a single pass over the tape. The audio and video time stamps and sequence numbers are aligned by the data striping tool as the video is stored to facilitate later media synchronization. The audio and video data is striped onto the disks using a user selected stripe size. Currently all of the video data on the server uses a 48 Kbytes stripe size since it provides good utilization at the peak load with 50% of the disk bandwidth delivering data to the media server components.

The media server components are a session control agent, the audio transmission agent, and the video transmission agent. The user connects to the session control agent on the server system and arranges to pay for the video service and network bandwidth. The session control agent then sets up the network delivery connections and starts the video and audio transmission agents. The audio and video transmission agents read the media data from the stripped disks and pace the transmission of the data onto the network. The video transmission agent scales the embedded bit-stream in real time by transmitting only the bit planes needed to reconstruct the selected resolution at the decoder. For example, a 320x240-pixel video sequence

with 8 bits of base and 4 bits of enhancement streams at 15 frames per second will transmit every other frame of video data with all 5 packets for each frame of the base and only two packets for the four most significant bits of the enhancement layer resulting, in 864 Kbits of bandwidth. The server sends the video and audio either for a point-to-point situation or a multicast situation.

The media player components are the software-based video decoder, the audio receiver, and a user interface agent. The decoder receives the data from the network and decodes it using lookup tables and puts the results onto the frame buffer. The decoder can run on any modern machine without significant CPU loading. The audio receiver loops reading data from the network and queuing up the data for output to the speaker. In the event of audio packet loss, the audio receiver attempts to ramp the audio level down to silence level and then back up to the nominal audio level of the next successfully received audio packet. The system performs media synchronization [27] to align the audio and video streams at the destination. End-to-end feedback is used in the on-demand case to control the flow. In the multicast case, the destinations are slaved to the flow from the server with no feedback. Figure 17 shows the block diagram of the system. The decoder can run on any modern machine without significant CPU loading. The entire scalable system makes collaborative video over heterogenous networks possible without any special-purpose hardware support.

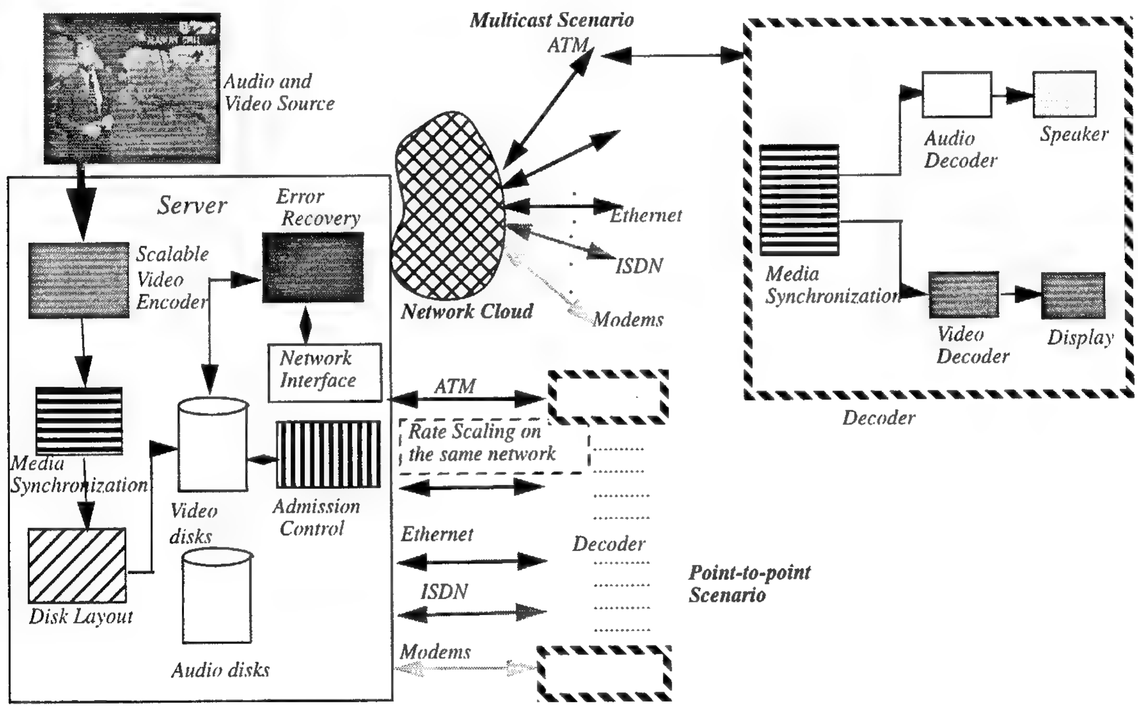


Figure 17. Block diagram of the scalable video delivery system.

4.7 Performance Results

A prototype of the scalable video system has been implemented. The video data rate varies from 19.2 Kbits/sec to 2 Mbits/sec, depending on the spatial and temporal requirements of the decoder and the

network bandwidth available. The PSNR varies between 31.63 dB to 37.5 dB. Table 6 gives the results for the decoding of 160x120 resolution video on a Sparc 20. It can be seen that the time required to decode the highest quality stream (8-bit index) at 160x120 resolution is 2.45 *ms* per frame (sum of lookup and packing time). This corresponds to a frame rate of 400 frames/sec. Similarly Table 7 gives the results for the decoding of 320x240 resolution video on a Sparc 20. It can be seen that the time required to decode the highest quality stream (8-bit base index and 8-bit first enhancement layer index) at 320x240 resolution is 7.76 *ms* per frame, corresponding to a frame rate of 130 frames/sec. Similarly Table 8 gives the results for the decoding of 640x480 resolution video on a Sparc Station 20, at 24.62 *ms* per frame, corresponding to a frame rate of 40 frames/sec

Table 9 shows the delay in servicing 160x120 resolution video. To deliver the highest quality stream (8-bit base) at 160x120 resolution takes 5.60 *ms* of CPU time and an average CPU load of 2% on a Sparc 20 workstation. The average disk seek time per frame is 16 *ms*. Similarly Table 10 shows the delay in servicing 320x240 resolution video. To deliver the highest quality stream (8-bit base and 8-bit enhancement layer) at 320x240 resolution takes 12.73 *ms* of CPU time and an average CPU load of 7% on a Sparc 20 workstation. The average disk seek time per frame is 18 *ms*.

Table 6. Results for 160x120 resolution video at the decoder.

No. of Bits of Lookup	PSNR (dB.)	Bandwidth as a function of frame rate (N) Kbps	CPU time per frame (ms)	Packing time per frame (ms)
4	31.63dB.	19.2 N	1.24 ms	0 ms
5	32.50 dB.	24 N	1.32 ms	0.52 ms
6	34 dB.	28.8 N	1.26 ms	0.80 ms
7	35.8 dB.	33.6 N	1.10 ms	1.09 ms
8	37.2 dB	38.4 N	1.18 ms	1.27 ms

Table 7. Results for 320x240 resolution video (8-bit lookup base).

No. of Bits of Lookup	PSNR (dB.)	Bandwidth as a function of frame rate (N) Kbps	CPU time per frame (ms)	Packing time per frame (ms)
2	33.72 dB.	48 N	6.01 ms	0.385 ms
4	35.0 dB.	52.8 N	6.04 ms	0.645 ms
5	35.65 dB.	62.4 N	6.05 ms	0.92 ms
6	36.26 dB.	67.2 N	6.08 ms	1.20 ms
7	36.9 dB.	72 N	6.04 ms	1.48 ms
8	37.5 dB.	76.8 N	6.09 ms	1.67 ms

Table 8. Results for 640x480 resolution video from interpolating 320x240-pixel images.

No. of Bits of Lookup	PSNR (dB.)	Bandwidth as a function of frame rate (N) Kbps	CPU time per frame (ms)	Packing time per frame (ms)
2	33.2 dB	48 N	22.8 ms	0.385 ms
4	34 dB	52.8 N	22.87 ms	0.645 ms
5	34.34 dB	62.4 N	23.14 ms	0.92 ms
6	34.71 dB	67.2 N	22.93 ms	1.20 ms
7	35.07 dB	72 N	22.90 ms	1.48 ms
8	35.34 dB	76.8 N	22.95 ms	1.67 ms

Table 9. Results for 160x120 resolution video at the disk server.

No. of Bits of Lookup	Bandwidth as a function of frame rate (N) Kbps	CPU time per frame (ms)	Seek-time (ms)	Avg. CPU Load
4	19.2 N	2.84 ms	16 ms	1%
5	24 N	3.67 ms	16 ms	1%
6	28.8 N	4.48 ms	14 ms	2%
7	33.6 N	4.92 ms	14 ms	2%
8	38.4 N	5.60 ms	16 ms	2%

Table 10. Results for 320x240 resolution video at the disk server.

No. of Bits of Lookup	Bandwidth as a function of frame rate (N) Kbps	CPU time per frame (ms)	Seek-time (ms)	Avg. CPU Load
2	48 N	10.47 ms	18 ms	6%
4	52.8 N	11.02 ms	16 ms	6%
5	62.4 N	11.55 ms	18 ms	6%
6	67.2 N	12.29 ms	20 ms	7%
7	72 N	12.55 ms	20 ms	7%
8	76.8 N	12.73 ms	18 ms	7%

References

[1] A. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low power CMOS digital design," *IEEE Journal f Solid-State Circuits*, pp. 685-691, April 1992.

[2] J. Woods, editor, *Subband image coding*, Kluwer Academic Publishers, 1991.

[3] M. Winzker, et al., "VLSI chip set for 2D HDTV subband filtering with on-chip line memories," *IEEE Journal of Solid-State Circuits*, Dec. 1993, vol. 28, no. 12, p. 1354-61.

[4] W. Lee, *Mobile Cellular Telecommunication Systems*, McGraw Hill, New York, 1989.

[5] B. Gordon and T. Meng, "A 1.2mW video-rate 2D color subband decoder," *1995 IEEE Inter. Solid-State Circuits Conference Digest of Technical Papers*, pp. 290-291, February 1995.

[6] T. Fischer, "A pyramid vector quantizer," *IEEE Trans. Inform. Theory*, vol. 32, pp. 568-583, July 1986.

[7] H. Tseng and T. Fisher, "Transform and hybrid transform/DPCM coding of images using pyramid vector quantization," *IEEE Trans. on Comm.*, vol. 35, pp. 79-86, January 1987.

[8] S. Purcell, "C-Cube CL550 image processor," *Proc. HOT Chip Symposium*, August 1990.

[9] E. Tsern and T. Meng, "Image coding using pyramid vector quantization of subband coefficients," *Proceedings of ICASSP 1994,* April 1994.

[10] A. Gersho and R. Gray, *Vector quantization and signal compression*, Kluwer Academic Publishers, Boston, 1992.

[11] N. Chaddha and T. H. Meng, "Psycho-visual based distortion measure for monochrome image compression," *Proc. SPIE Visual Communications and Image Processing*, November 1993.

[12] A. Hung and T. Meng, "Error resilient pyramid vector quantization for image compression," *Proc. of International Conference on Image Processing*, November 1994.

[13] G. Clark, Jr., and J. Cain, *Error-correction coding for digital communications.* Plenum Press, New York, 1981.

[14] S. Lin and D. Costello Jr., *Error control coding: fundamentals and applications*, Prentice Hall, Englewood Cliffs, 1983.

[15] E. Tsern, T. Meng, and A. Hung, "Video compression for portable communication using lattice vector quantization of subband coefficients," *IEEE Workshop on VLSI Signal Processing*, pp. 444-452, October 1993.

[16] D. Taubman and A. Zakhor, "Multi-rate 3-D subband coding of video," *IEEE Trans. IP.*, Vol. 3, No. 5, pp. 572-588, Sep. 1994.

[17] J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. SP.,* Vol. 41, No. 12, pp. 3445-62, Dec. 1993.

[18] A. Gersho and R. Gray, *Vector quantization and signal compression,* Boston, MA: Kluwer Academic Pub., 1991.

[19] P. Chou, T. Lookabaugh and R. Gray, "Optimal pruning with applications to tree-structured source coding and modelling," *IEEE. Trans. IT.,* Vol. 35, pp. 299-315, 1989.

[20] E. Riskin and R. Gray, "A greedy tree growing algorithm for the design of variable rate quantizers," *IEEE Trans. Signal Processing,* Vol. 39, pp. 2500-07, 1991.

[21] N. Chaddha, W. Tan and T. Meng, "Fast vector quantization algorithms for color palette design based on human vision perception," to appear in *IEEE Trans. Image Processing*, 1995.

[22] N. Chaddha, "An efficient algorithm for scalable video compression with software only decode," *Technical Report Sun Microsystems*, September 1994.

[23] F. Tobagi, et al., "Streaming RAID: A disk array management system for video files," *Proc. ACM Multimedia*, 1993.

[24] M. Miller, "Extending markets inward," *Proc. Bionomics Conference*, San Francisco, Oct. 1994.

[25] N. Chaddha and T. Meng, "Psycho-visual based distortion measures for image and video compression", *Proc. of Asilomar Conference on Signals, Systems and Computers*, Nov. 1993.

[26] N. Chaddha, P. Chou, and T. Meng, "Scalable compression based on tree-structured vector quantization of perceptually weighted generic block transformations," submitted to *International Conference on Image Compression*, January 1995.

[27] J. D. Northcutt and E.M. Kuerner, "System Support for Time-Critical applications," *Proc. NOSSDAV' 91*, Germany, pp. 242-254.

5.0 Sponsored Publications in 1994 and 1995

[1] T. Meng, B. Gordon, E. Tsern, and A. Hung, "Portable video-on-demand in wireless communication," invited paper, to appear in the *Proceedings of IEEE*, Vol. 83, No. 4, April 1995.

[2] B. Gordon and T. Meng, "A 1.2mW video-rate 2D color subband decoder," *1995 IEEE Inter. Solid-State Circuits Conference Digest of Technical Papers*, pp. 290-291, February 1995.

[3] W. Tan and T. Meng, "A low-power high performance polygon renderer for computer graphics", to appear in *Journal of VLSI Signal Processing,* Vol. 9, No. 4, pp. 233-255, May 1995.

[4] A. Hung and T. Meng, "A comparison of fast inverse discrete cosine transform algorithms," *Multimedia Systems Journal*, No. 5, pp. 204-217, February 1995.

[5] C. Portmann and Teresa H.-Y. Meng, "Metastability in CMOS library elements in reduced supply and technology scaled applications," *IEEE Journal of Solid-State Circuits*, Vol. 30, No. 1, pp. 39-46, January 1995.

[6] S. Hemami and T. Meng, "Transform coded image reconstruction exploiting interlock correlation," to appear in *IEEE Trans. on Image Processing*, 1995.

[7] N. Chaddha, W. Tan and T. Meng, "Fast vector quantization algorithms for color palette design based on human vision perception," to appear in *IEEE Trans. Image Processing,* 1995.

[8] B. Wei and T. Meng, "A parallel decoder of programmable Huffman codes," to appear in *IEEE Trans. on Video Technology*, 1995.

[9] A. Hung and T. Meng, "Error resilient pyramid vector quantization for image compression," *Proc. of International Conference on Image Processing*, November 1994.

[10] B. Wei and T. Meng, "A parallel decoder of programmable Huffman codes," *Proc. of International Conference on Image Processing*, November 1994.

[11] B. Gordon, N. Chaddha and T. Meng, "A low-power multiplierless YUV to RGB converter based on human vision perception", *Proc. of 1994 IEEE Workshop on VLSI Signal Processing*, October 1994.

[12] N. Chaddha, W. Tan and T. Meng, "A real-time color quantizer trainer/encoder," *Proc. of the 28th IEEE Asilomar Conference on Signals, Systems and Computers*, October 1994.

[13] N. Chaddha, W. Tan, and T. Meng, "Color quantization of images based on human vision perception," *Proceedings of ICASSP 1994*, April 1994.

[14] E. Tsern and T. Meng, "Image coding using pyramid vector quantization of subband coefficients," *Proceedings of ICASSP 1994,* April 1994.

[15] B. Gordon and T. Meng, "A low power subband video decoder architecture," *Proceedings of ICASSP 1994,* April 1994.

[16] N. Chaddha, T. Meng and et al., "Scalable baseline-JPEG compression algorithm for monochrome images", *Proc. IEEE International Conference on Multi-Media Systems*, May 1994.

[17] M. Engels and T. Meng, "Rapid prototyping of a real-time video encoder," *Proc. of IEEE International Workshop on Rapid System Prototyping*, June 1994.

[18] A. Hung and T. Meng, "Multidimensional rotations for quantization," *Proc. Data Compression Conference*, April 1994.

[19] A. Hung and T. Meng, "Statistical inverse discrete cosine transforms for image compression," *SPIE Proc. Electronics for Imaging*, February 1994.

6.0 Project Personnel

The personnel supported during the past six months are listed below.

Faculty: Teresa H. Meng, principle investigator and research advisor for the students listed below.

Students: Navin Chaddha, Ben Gordon, Andy Hung, Clem Portmann, and Ely Tsern.

Secretary: Lilian Betters.

Point of contact: Teresa H. Meng, CIS 132, Stanford University, Stanford, CA. 94305.

Phone number: (415) 723-2252, FAX number: (415) 725-6278, email: meng@mojave.stanford.edu.

Appendix A: Video Compression Algorithms for Portable Applications

Because our portable video-on-demand system is to be embedded in a wireless communication environment, the first step in designing a suitable compression algorithm is to analyze the wireless channel characteristics. Wireless channels, depending on the channel modulation techniques used, display a wide range of error patterns. Experiments with mobile receivers and transmitters show fades exceeding 10 *dB* about 20% of the time, and at 15 *dB* (measuring limit) up to 10% of the time [4], causing bursty bit errors in the received date stream. Bursty bit errors are usually handled by interleaving the transmitted data stream over a certain period of time so that if consecutive bit errors occur (as is typical with bursty bit errors) in the received data stream, the error pattern appearing in the decoded data stream will resemble random bit errors. As the time interval of deep fades is usually within tens of milliseconds while the data rate of compressed video is between 500 *Kbps* and 1.5 *Mbps*, interleaving can be easily accommodated with a data buffer, the size of which is in the range of tens of kilobits. We will therefore consider the effects of channel error to be random bit errors in the received data stream.

To reduce the effects of channel random bit errors, error-correcting codes are usually used. With the knowledge of a target channel bit error rate (BER) and the channel's signal-to-noise ratio, error-correcting codes can often be very effective. However, for wireless and mobile channels that experience very low BERs most of the time but with intermittent severe channel degradation occasionally, error-correcting codes may not be the best solution. On the one hand, under low distortion conditions (low BERs), error-correcting codes add unnecessary overhead to transmission bandwidth, which could have been used for transmitting compressed video data to improve video quality. On the other hand, under severe distortion conditions in which the BER exceeds the designed capacity, error-correcting codes may actually introduce more decoded errors than received ones. On top of all this, error-correcting codes require additional hardware at the decoder unit, making the design of our portable decoder a more difficult task.

Our approach to error resistance is to embed error-resiliency into the compression process so that if channel error does occur, the effects of it will be a gradual degradation of video quality and the best possible quality will be maintained at all BERs. In this way, we do not pay a high premium in bandwidth when protection against error is not needed, but still deliver reasonable-quality video when the channel BER is extremely high. Because our goal is to transmit compressed video data -- and human vision is fairly fault-tolerant -- exact binary reproduction at the decoder is not necessary (which is an assumption made in using error-correcting codes). As long as the effects of error are localized and do not cause catastrophic loss of image sequences, a robust error-resilient compression algorithm without resorting to error-correcting codes for data protection can be the solution, our best compromise among the goals of consistent video quality, minimal transmission bandwidth, and low-power implementation.

To achieve these goals, our compression algorithm utilizes lattice vector quantization (VQ), of which pyramid vector quantization (PVQ) is a special case, together with subband decomposition. Lattice VQ provides several distinct advantages. First, it is a fixed-rate code, which results in hardware simplicity and prevents catastrophic error propagation. Second, because of its regular lattice structure, lattice VQ allows for simple real-time decoding and encoding. Third, when optimized for the statistics of image data, lattice

VQ provides excellent rate-distortion performance for moderate to high bit rates, achieving the compression performance of entropy codes asymptotically [6] [7].

Subband decomposition offers additional compression by decomposing the information energy of image data into several frequency bands. Unlike the DCT used in standard compression, the subband approach does not introduce blocking artifacts. Subband decompressed images are therefore usually considered to be more visually pleasing. In addition, the hierarchical nature of subband decomposition allows for flexibility in bit allocation, with different bit rates assigned to different subbands based on information content, visual importance, and subband size. Finally, subband decomposition provides many possible algorithm trade-offs to achieve a low-power implementation.

6.1. Subband Decomposition

As shown in Figure 18, subband filtering decomposes each video frame into various subbands by passing the frame through a series of 2-D low-pass and high-pass filters. Each level of subband decomposition divides the image into four subbands. We can hierarchically decompose the image by further subdividing each subband. We refer the pixel values in each subband as subband coefficients.

The filtering process decorrelates the image information and compacts the image energy into the lower frequency subbands, a natural result of the fact that most image information lies in low spatial frequencies. The information in high spatial frequencies, such as edges and small details, lies in the higher frequency subbands, which have less energy (pixel variance). We achieve compression by quantizing each of the subbands at a different bit rate according to the amount of energy and the relative visual importance in each subband.

The choice of encoding and decoding filters is determined by their ability to accurately reproduce the original image and cancel aliasing between subbands. Examples are quadrature mirror filters (QMF) which have near-perfect reconstruction and wavelet filters with perfect reconstruction in magnitude, though with a non-linear phase response [8].

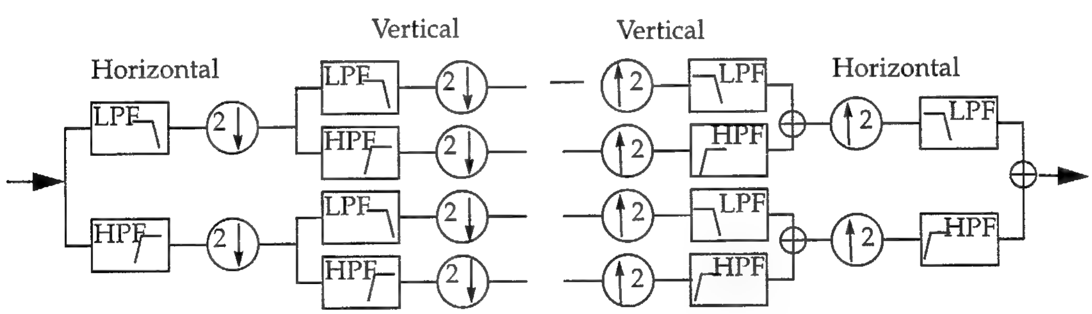


Figure 18. One level of 2-D subband filtering.

We apply four levels of subband decomposition in our compression scheme, with different bit allocation patterns for the three YUV components at different rates. For decoding, the four lowest-level subbands are each upsampled and filtered, then summed together to form reconstructed higher-level subbands. This process is repeated through all four levels until the final image is formed.

6.2. Pyramid Vector Quantization

PVQ is a lattice vector quantization technique that groups data into L-dimensional vectors and scales them on to an L-dimensional pyramid surface [6]. An example is shown in Figure 19, where L is the vector dimension and K is an integer defined as the *pyramid radius*. Figure 19 also illustrates a pyramid surface for $L = 3$ and $K = 4$.

Pyramid surface defined by:

$$\sum_{i=1}^{L} |x_i| = K$$

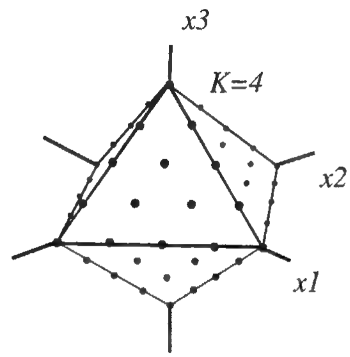


Figure 19. A 3-D pyramid with pyramid radius K equal to 4.

The lattice points on the pyramid form the VQ codebook, with each lattice point assigned a binary codeword index. The pyramid radius and the vector dimension determine the codebook size and thus the coding rate.

The basic quantization steps of PVQ are (1) scalar-quantize the *vector radius*, r, defined as the absolute norm of the vector to be coded, (2) scale the vector onto a pyramid of a constant radius, by multiplying each vector element by K/r, and (3) quantize the scaled vector to the nearest lattice point on the pyramid. Both the pyramid lattice index and the scalar index are then transmitted. This version of PVQ is generally referred to as product PVQ [6].

Because the pyramid structure is regular, encoding and decoding codewords can be done with a simple formula, which has computational complexity linearly proportional to the vector dimension and coding rate. Thus, PVQ eliminates the need for codebook storage and training found necessary in most VQ designs [10], effectively allowing for very large codebooks unrestricted by the size of physical memory. As discussed in Sections 2, read/write accesses to large memories consume far more power than arithmetic computation, often by a factor of 10 for on-chip memory accesses, not to mention the power needed for off-chip memory accesses. PVQ therefore allows for the reduction of power-consuming memory accesses in favor of arithmetic computation, offering low-power encoding and decoding at video rates.

6.3. Selection of Vector Dimensions

The PVQ vector dimension greatly impacts the performance and implementation of PVQ on both its hardware complexity and error-resiliency capability. While larger vector dimensions achieve better compression performance, using a smaller vector dimension significantly improves error resiliency by localizing the effect of channel bit errors [9].

One major benefit of using small vector dimensions is significant reduction in hardware, a result of smaller codeword indices and correspondingly smaller datapath widths and memory. The PVQ encoding and decoding algorithm requires table look-ups of pre-computed factorial calculations and index offset values for locating the nearest lattice point of a vector. The size of total memory required for these tables is $LK_{max}bpp_{max} + (L-3)\left(K_{max}-2\right)bpp_{max} + L\left(K_{max}-1\right)\left\lceil \log\left(Lbpp_{max}\right)\right\rceil$, where L is the vector dimension, K_{max} is the maximum pyramid radius, and bpp_{max} is the maximum bit rate expressed in bits per pixel. With a larger vector dimension, a larger radius K is required to maintain the same bit rate, which increases the size of memory needed to store the tables significantly. For example, for $L = 4$ and a maximum bit rate of 5 bits/pixel ($K = 73$), the memory size is 8,700 bits. For $L = 16$ at the same maximum bit rate of 5 bits/pixel ($K = 123$), the required memory size is 295,904 bits, a factor of 34 times more memory. However, to code subband coefficients in high frequency subbands where the allocated bit rates are relatively low, we can afford larger dimensions for better compression capability. Our PVQ coder uses a vector dimension of one (PCM) on the lowest frequency subband, up to a dimension of 32 on the highest frequency subband.

6.4. Compression Performance

To demonstrate the compression performance of this fixed-rate subband/PVQ algorithm, we applied it to the USC database images *Lena*, *Lake*, *Couple*, and *Mandrill* at various compression ratios. First our subband/PVQ compression exceeds previously reported PVQ results for similar images. To provide a relative measure of performance, we compare our fixed-rate, intra-frame technique to JPEG, which is an intra-frame, DCT-based algorithm using the run-length and Huffman codes to achieve a high degree of compression. As shown in Figure 20, it outperforms JPEG at all bit rates of interest (JPEG with scaled default quantization matrices and custom-generated Huffman tables), without requiring the use of variable-rate entropy codes [12]. Though the signal-to-noise ratio performance may not be a distortion metric faithfully representing human visual perception, our study on image quality assessment based on psycho-vision also suggested that the subband/PVQ algorithm delivers images of better quality than JPEG at the same compression ratio [11].

6.5 Error-Resilient PVQ Indexing

As discussed earlier, bursty bit errors can often be modeled by random bit errors when data is interleaved for transmission. Because our subband/PVQ algorithm guarantees a constant compressed bit rate as determined by subband bit allocation, random bit errors do not corrupt pixel position information, and error effects will not propagate over vector boundaries. For PVQ, random bit errors on a transmitted codeword index may cause each element (one element representing a subband coefficient) of the decoded vector to deviate from its original value, sometimes to its negative value if the index enumeration, which assigns a binary bit pattern to represent each lattice point, is not well designed. In the following, we describe an index enumeration scheme that minimizes the effects of random bit errors on decoded image pixels [12].

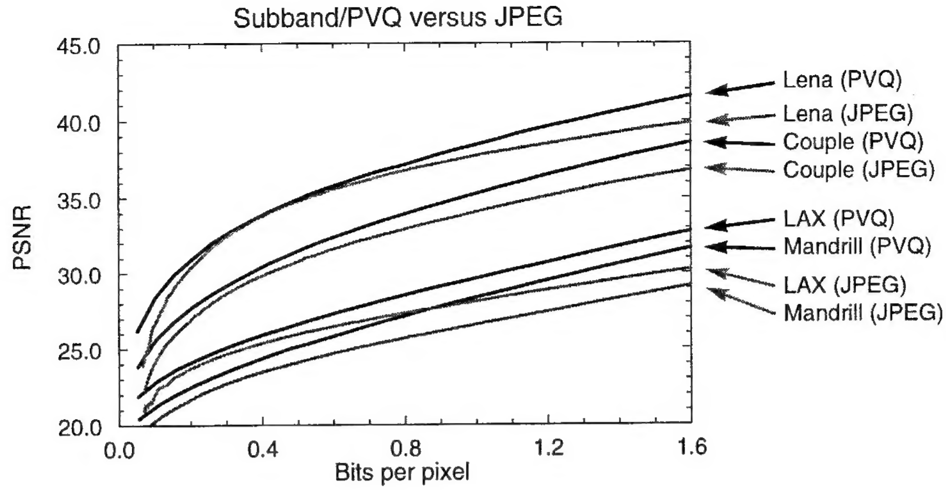


Figure 20. Compression performance in peak signal-to-noise ratio (PSNR) of subband/PVQ versus JPEG on USC database images *Lena, Couple, LAX*, and *Mandrill.*

As reported in the July 1993 report, our indexing scheme codes each vector by four characteristics: (1) the number of non-zero vector elements, (2) the sign pattern, which describes the signs of the non-zero vector elements, (3) the vector pattern, which describes the positions of the non-zero vector elements, and (4) the vector shape, which describes the magnitudes of the non-zero vector elements. This method codes each characteristic separately, then combines them all in an implicit product code. Separating these characteristics in a product code helps limit the effect of bit errors.

To form a codeword index for a vector, we first enumerate the vector pattern, multiply that by the number of possible shapes, and then add the enumerated vector shape. Finally, we append the sign bits to the least significant positions of the codeword index. The decoding algorithm requires at most $(LK+5)$ memory fetches, $(LK+8)$ compares, $(LK+5)$ subtracts, and one divide.

The enumeration is channel-error resistant because the sign bits form a binary product code -- a single bit error in the sign bits only causes a sign change in a single pixel without affecting other pixels. Furthermore, the pattern information rests in only a few of the most significant bits of the product code, while the shape information composes the majority of the codeword. Thus, single bit errors that occur in most bits only affect the shape (pixel values) without changing its sign; only bit errors that occur in the few most significant bits would corrupt both the pattern and the shape.

6.6. Error Resiliency Performance

We now evaluate our compression scheme based on its error-resiliency properties. To make the comparison fair, we used JPEG's resynchronization intervals of 6 and added error-correcting codes on top of JPEG-coded data, as would be necessary if JPEG was used in wireless transmission. To test the effectiveness of different error-correcting codes in improving the image quality at various bit error rates (BERs), we selected three commonly used error-correcting codes: a (127, 120) *Hamming code*, which incurs a bandwidth overhead of 10% to the overall bit rate and is capable of correcting one bit error in consecutive 127 bits, a (73, 45) *Weldon difference-set* error-correcting code with a threshold decoding

scheme, which offers good protection (Hamming distance of 10) and incurs a bandwidth overhead of 60% to the overall bit rate [13], and a (2,1,6) *NASA planetary* convolutional code, which offers a high degree of error correction capability but requires Viterbi decoding at the decoder and doubles the overall bit rate [14]. For subband/PVQ, we apply a simple 3-bit repetition code on the most significant bit (MSB) and the next MSB of each pixel in the lowest-frequency subband. Because the 4-level subband decomposition effectively compacts most image energy into the lowest-frequency subband, the size of which is only $^1/_{256}$ of the total image, this simple DC protection scheme incurs only a bandwidth overhead of 0.016 bit/pixel to the overall bit rate. In the following simulation the subband/PVQ algorithm actually uses a slightly lower bit rate than JPEG, even with the added DC protection. The simulation was conducted using image *Mandrill* compressed at 12:1 or 0.66 bit/pixel, a rate at which both JPEG and the subband/PVQ algorithms give good compression performance.

Figure 21 graphically shows the performance of our algorithm under noisy channel conditions for the *Mandrill* image. Under most error conditions, our subband/PVQ algorithm performs better than JPEG with or without protection, as expected from the fact that our algorithm outperforms JPEG even without channel error (as shown in Figure 20). Over a wide range of low BERs, the subband/PVQ algorithm delivers images of higher quality because the whole channel bandwidth can be used for transmitting compressed video data. On the JPEG with error correction curves, we note that error-correcting codes do an excellent job of eliminating bit errors, but start to fail at past BER of 10^{-2}. Once the BER increases past this point, catastrophic errors occur, causing severe block loss and rapid drop in peak signal-to-noise ratio (PSNR) performance. Our fixed-rate scheme, however, maintains a gradual decline in quality, even under severe BER conditions. This gradual degradation performance makes the subband/PVQ scheme well suited for situations where image quality must be maintained under deep channel fades and severe bit loss, a characteristic of wireless channels.

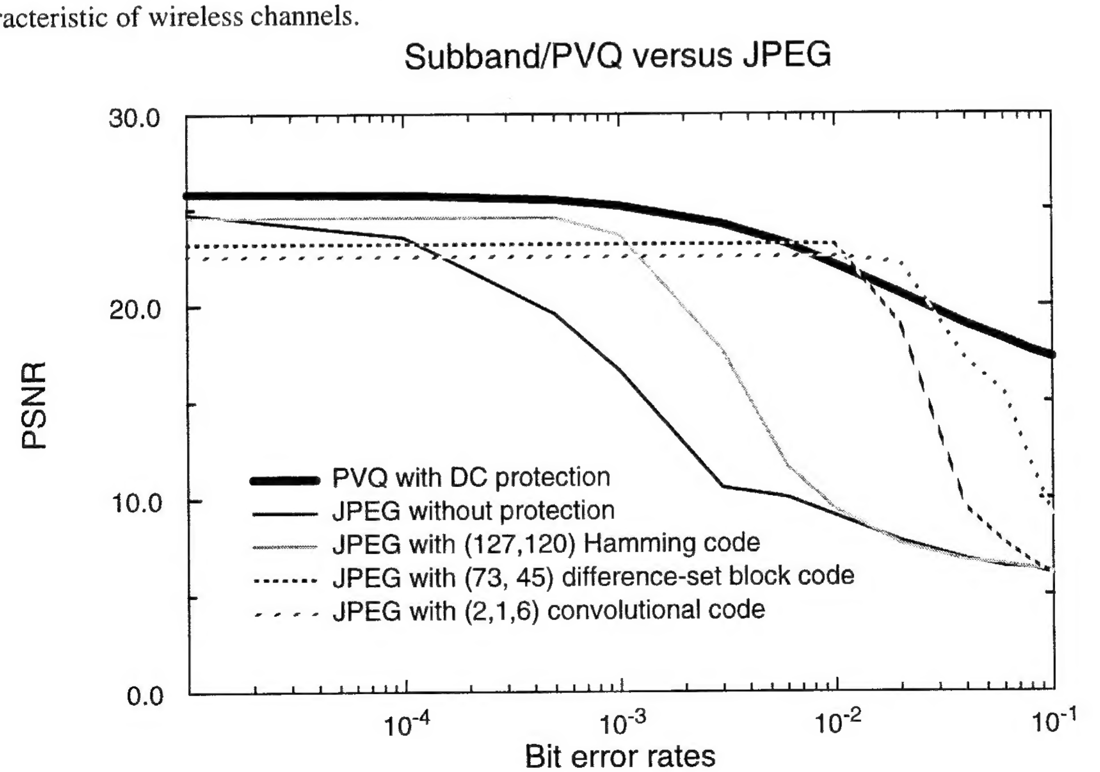


Figure 21. Peak signal-to-noise ratios (PSNR) vs. bit error rates (BER) for subband/PVQ and JPEG with error-correcting codes.

To show the effects of bit errors on decompressed images, we used image *Mandrill* compressed at 0.66 bit/pixel as an example. We corrupted the transmitted data at BERs of 10^{-3} and 2×10^{-2}. In order to prevent error propagation in decoding JPEG-coded images (as a result of using entropy codes), we used the (73,45) error-correcting code mentioned above, reducing the effective bandwidth for the image data by approximately 40%. Figure 22 shows decompressed images under such channel conditions. At the BER of 10^{-3} (Figure 22 (a) and (c)), no noticeable artifact caused by channel error is present, indicating that the error-correcting code is quite effective in correcting random bit errors at this rate. The JPEG image, however, displays lower quality because less bandwidth was available for transmitting compressed image data. At the BER of 2×10^{-2}, the subband/PVQ image (Figure 22 (b)) can still be recognizable with reasonable quality, though bit errors cause ringing artifacts, locally distorting the image. In the JPEG image (Figure 22 (d)), bit errors become very noticeable when synchronization bits are corrupted, causing total block loss.

As for the quality of compressed *video*, when the BER approaches 10^{-3}, the random loss of blocks in JPEF-coded video sequences becomes evident, causing a flickering that makes it difficult to view details of the video. With the subband/PVQ video, increasing bit errors cause increased blurriness and wavering artifacts, but most of the video detail remains distinguishable for BERs past 10^{-2}.

We attribute the resiliency of our algorithm to the following factors:

1. The fixed-rate nature of our algorithm prevents error propagation and eliminates the need for resynchronization.
2. Subband decomposition compacts most image information to the lowest-frequency subband, where PCM is used to encode each pixel, limiting the effects of bit errors to a small region.
3. Error-resilient index enumeration helps improve image quality by up to 3 dB compared to random indexing under high BER conditions [5].

In summary, we demonstrate an entirely fixed-rate compression scheme based on subband/PVQ that exceeds the compression performance of the variable-rate JPEG standard and delivers consistent image quality over a wide range of channel error conditions.

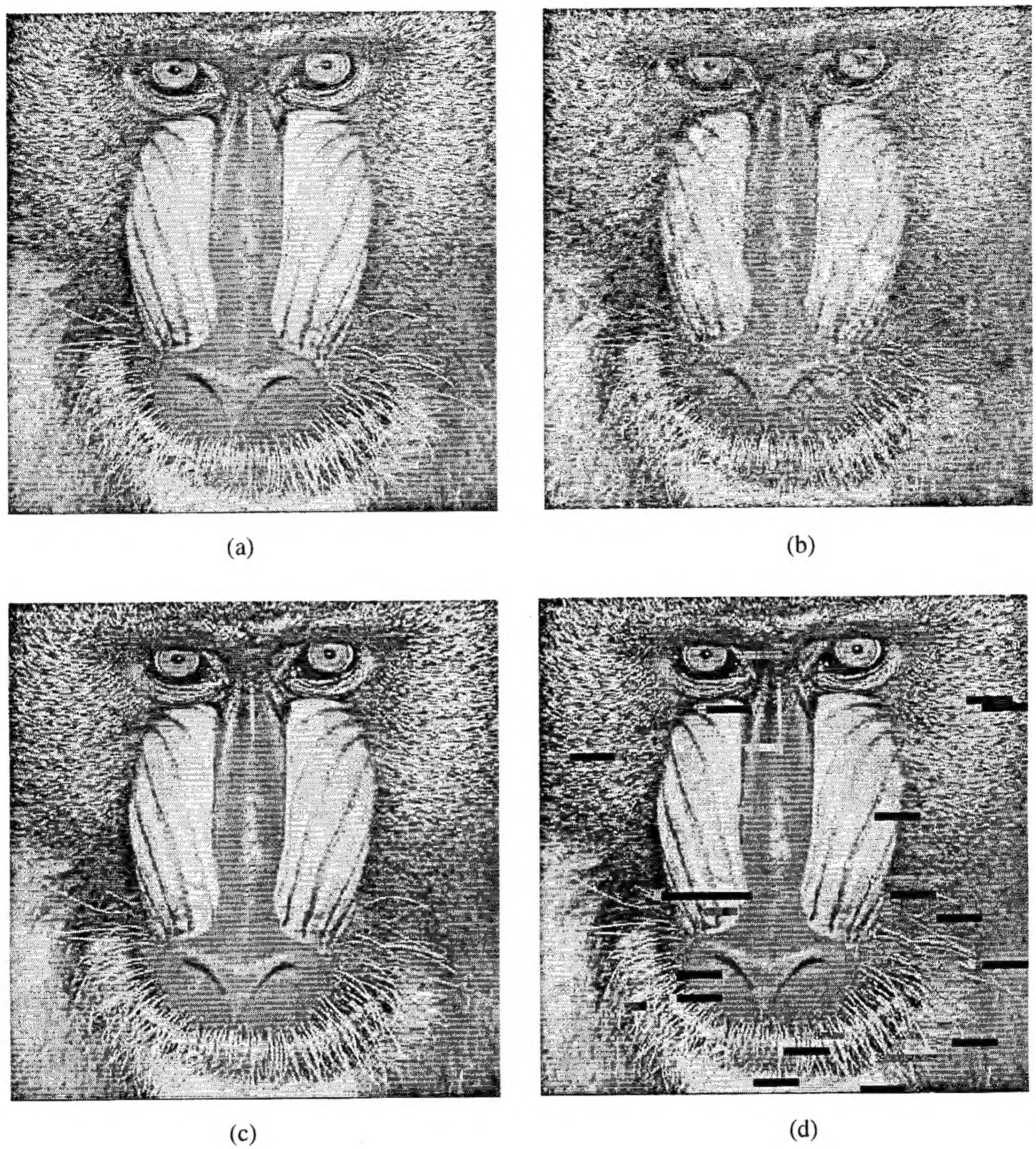


Figure 22. Compressed *Mandrill* at 12:1 (0.66 bit/pixel) with BERs at 10^{-3} and 2×10^{-2}: (a) and (b) were encoded using the subband/PVQ algorithm, while (c) and (d) were encoded using JPEG with the (73, 45) block code.